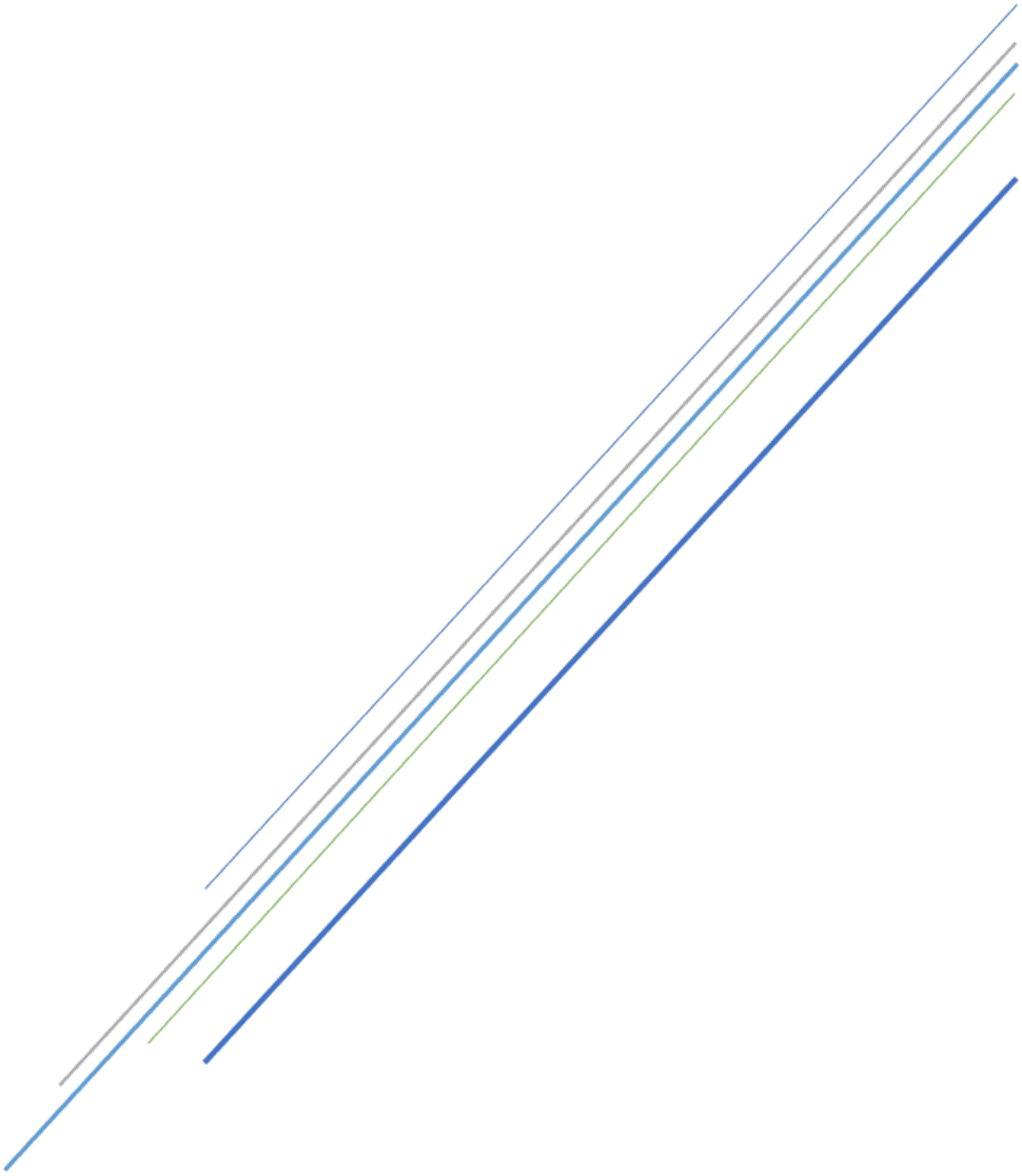
SQL2022

Server Documentation



Contact Information

AI-DBA Software Inc.
1500 West Georgia St. Vancouver, British Columbia V6G2Z6, Canada.
www.AI-DBA.net

Table of Content

Action Required by Administrator

Database administrator is required to pay attention to the following recommendations.

Type: Important

Title: Update Statistics

Description: Updating statistics ensures that the query optimizer has accurate information about the distribution of values in a column or columns used by queries.

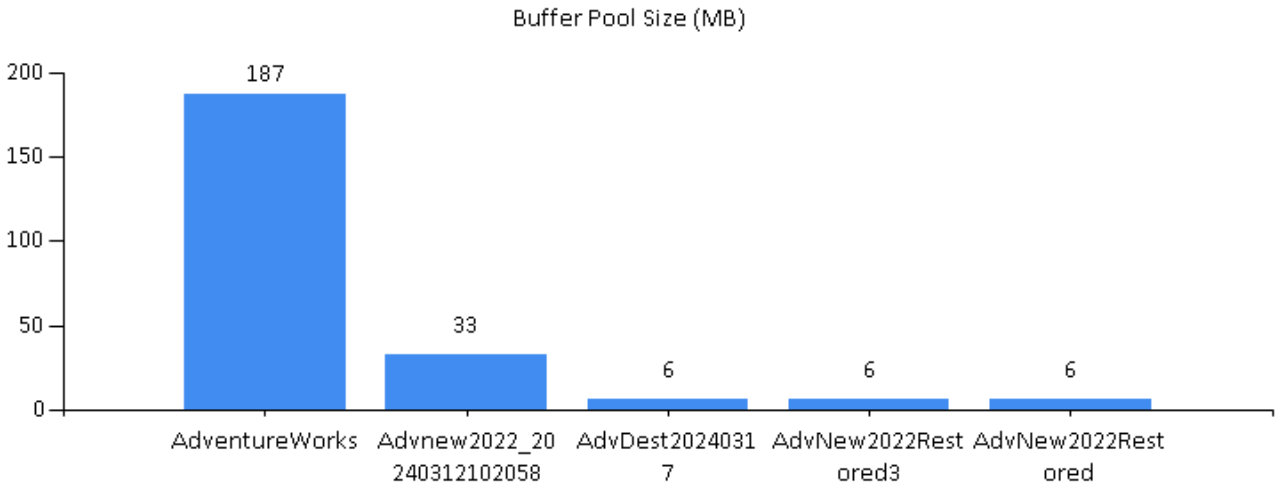
When SQL Server executes a query, it uses statistics to estimate the number of rows that match certain predicates and make decisions on which indexes or execution plans to use. If statistics are outdated, meaning they do not reflect the current state of data distribution, then the estimates made by the query optimizer may be incorrect.

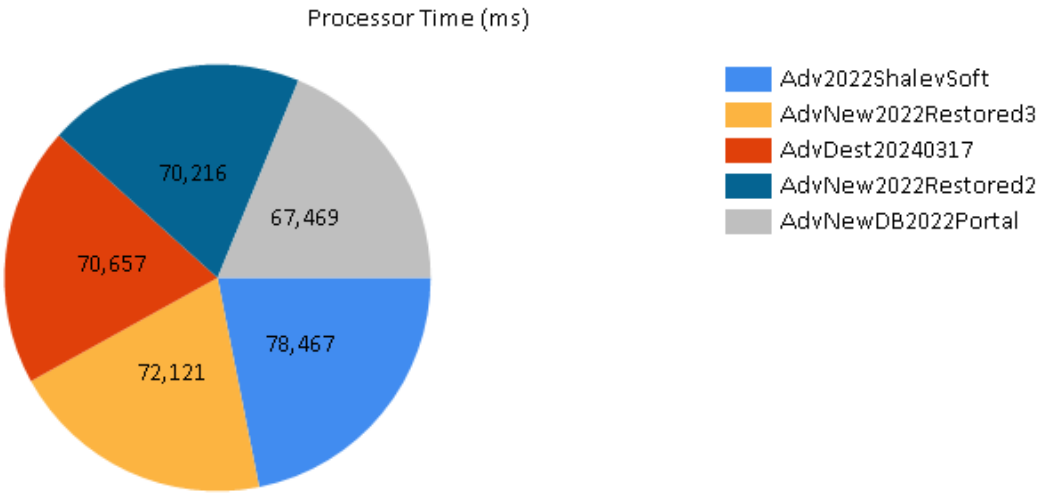
By updating statistics, you ensure that these estimates are as accurate as possible. This helps the queries to compile with up-to-date statistics so that they can generate optimal execution plans based on current data distribution. This can lead to improved query performance and overall system efficiency.

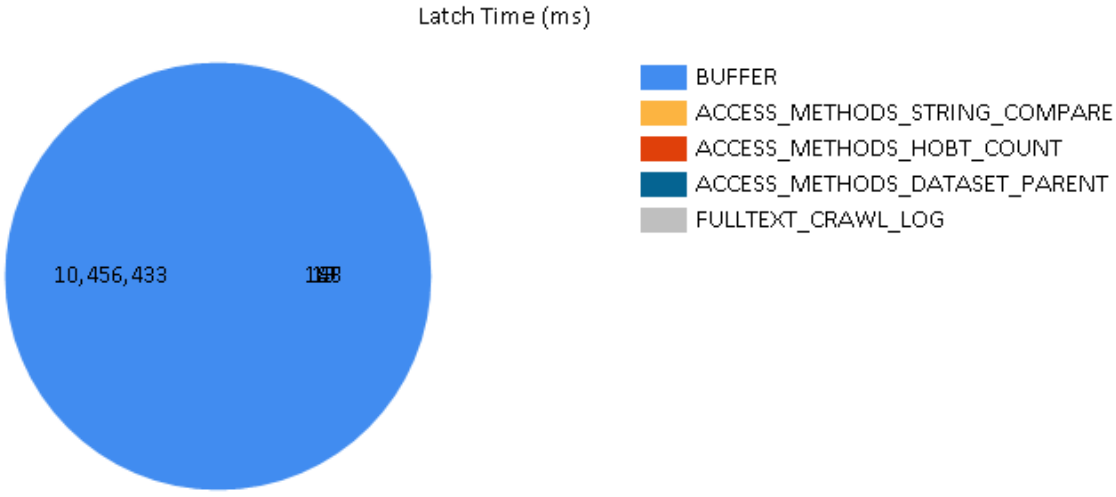
InfoMessage:

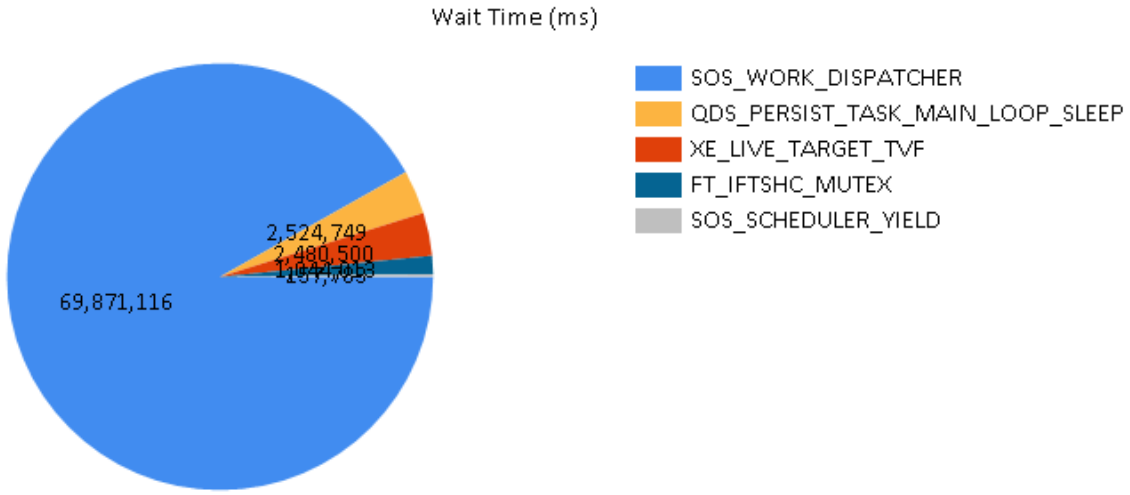
SQL Server Instance Health Overview

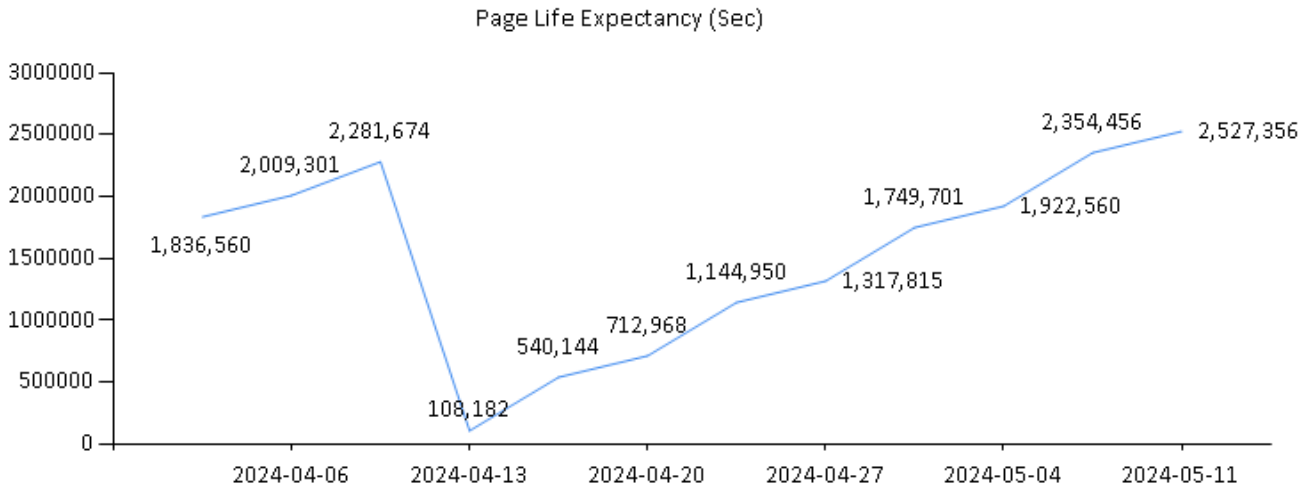
Provided to give you a glance of SQL Server instance health status.









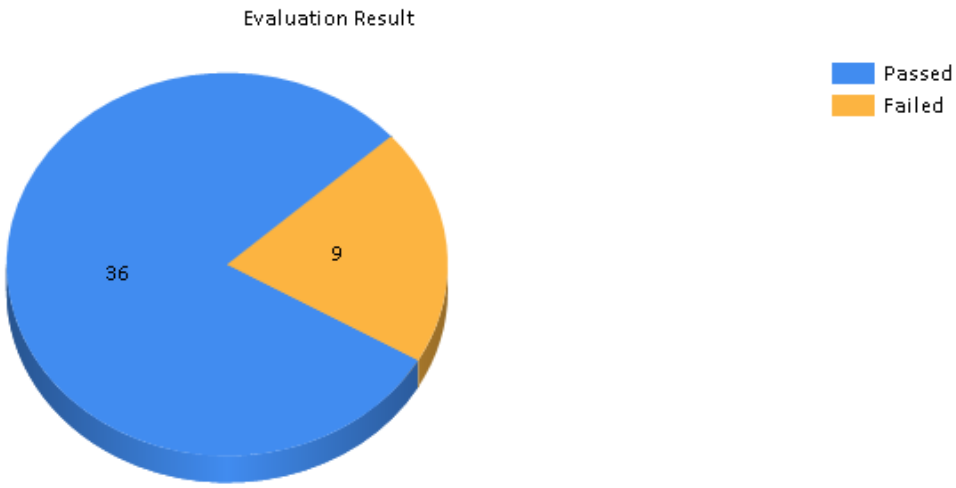


Physical Memory (MB)	Plan Cache (MB)	Pressure %	Description
8,190.0	409.0	153.3	Plan Cache - Local Memory Pressure

Processor Wait Signal	Resource Wait Signal	Description
6.3	93.7	Signal Waits later 15-17% is usually a sign of CPU pressure

SQL Server Instance Evaluation and Scores

Administration	Maintenace	Performance	Security	High Availability
4.8 / 5	4.8 / 5	3.5 / 5	3.5 / 5	5.0 / 5



Category	Title	Result
Administration	Database Full Recovery Model	Passed
Administration	Database File Federation	Passed
Administration	Database Table Partitioning for Tables >2mil Records	Passed
Administration	Database File IO Balancing	Passed
Administration	Database Full Backup in Past 7Days	Failed
Administration	Database Log Backup in Past 12Hours	Passed
Administration	Agent Job Failures for <10 Times	Passed
Administration	Database Auto Growth For <10 Times	Passed
Maintenance	Database Compression for Tables with >1mil Records	Passed
Maintenance	Database with No Suspected/Corrupted Pages	Passed
Maintenance	SQL Server with No Dump File.	Passed
Maintenance	Database Consistency Check in Past 14Days	Passed
Maintenance	Database Log Shipping Error Free in Past 7Days	Passed
Maintenance	Database Mirroring Error and Delay Free in Past 7Days	Passed
Maintenance	Moderated Database Transaction Log Fragmentation	Failed
Maintenance	Upgraded to the Least Support Database Engine Version	Passed
Maintenance	Database Engine with No Severe (Severity >16) Error	Passed
Maintenance	Operating System with No Severe Error	Passed
Security	Database Objects are Owned by non-DBO	Passed
Security	Database with No Orphaned Users	Passed

Category	Title	Result
Security	Login Authentication with <5 Failures	Failed
Security	>25 Active Sessions with SysAdmin Privileges	Failed
Security	SQL Server services with NT Service Account	Failed
Security	Login Brutal Attack Protection	Passed
Security	Database Engine TCP Listening Port Protection	Passed
Performance	<15ms IO Stall for Database Files	Failed
Performance	Query Design with Minimum Processor Intensive Functions	Failed
Performance	Trivial and Moderated Execution Plans	Passed
Performance	In-Memory with Utilized Hash Buckets	Passed
Performance	Database Index Design with Best Practices	Failed
Performance	Database Indexes with Low Fragmentation	Passed
Performance	Queries with an Index Support (No Missing Indexes)	Passed
Performance	Queries without an Index Influence (With Index Clause)	Passed
Performance	Database Task Request with Zero Pending Disk IO	Passed
Performance	Trivial Plans with Low Generation Cost	Passed
Performance	Low Database Deadlock and Blocking Occurrence in Past 7Days	Passed
Performance	Low Internal Pressure in Plan Cache Buckets	Passed
Performance	Low External Pressure in Plan Cache Buckets	Failed
Performance	Low Session Blocking in the Past 7Days	Passed
Performance	Low Pressure on Processor Resource	Passed
Performance	Low Pressure on System Memory Resource	Passed
Performance	Low Wait on Disk, Memory, Processor and Network Resources	Passed
High Availability	All Cluster Nodes are Up and Running	Passed
High Availability	The AlwaysOn Availability Group Data Loss and Recovery is Less Than 120Seconds	Passed
High Availability	The Database Mirroring Session is Up and Running	Passed

Hardware Specification

Property Item	Value
Physical Processors	1
Logical Processors	2
Hyperthread Ratio	2
Physical Memory (MB)	8,190
Committed Memory (MB)	3,265
Machine Type	Virtual

Volume	Capacity (MB)	Available Space (MB)	Free Space %	Bus Type	Media Type	IO Stall (MS)
C:\	129,481.0	41,175.0	31.8	SAS	Unspecified	6.2
G:\	65,405.0	58,472.1	89.4	SAS	Unspecified	7.8
H:\	65,405.0	60,565.0	92.6	SAS	Unspecified	58.1

Server Insight

The server insight content is based on the collected telemetry data and analyzed by AI engine.

* Last Update: 2024-05-11 10:32:38 UTC
Based on the provided data, we can analyze the storage space and gain insight into the data growth rate and potential storage capacity needs.

The storage space information is as follows:
- Date: 2024-03-30 10:30:03
- Volume C:\ has a free space percentage of 49.0%.
- Volume H:\ has a free space percentage of 94.2%.
- Volume G:\ has a free space percentage of 89.8%.

Insights:
1. The volume C:\ is consistently at around 49.0% free space, indicating that there may not be significant data growth or increased storage needs for this volume.

2. Both volumes H:\ and G:\ have high percentages of free space (94.2% and 89.8%, respectively), suggesting that they have ample room for future data growth.

3. However, since we only have one timestamped snapshot of the storage spaces without historical trends or additional information about the database size or activity, it's challenging to determine the exact growth rate and predict future storage capacity needs accurately.

To better gauge your SQL Server instance's overall data growth rate and identify specific tables/objects contributing to that growth, it would be beneficial to track and monitor daily or periodic snapshots over an extended period along with other performance metrics such as query execution time, disk I/O usage, transaction logs usage, etc., which are essential factors in determining when you might need to consider increasing your storage capacity.

Regarding long-running queries analysis:
The top three long-running queries in terms of average elapsed time (in seconds) are provided:

1) Average elapsed time: 0.23 seconds
Query: SELECT REPLACE(perms.class_desc...)
This query appears to relate to permissions' retrieval from various system objects like schemas, principals (users), assemblies, types, service contracts/consumers/producers/bindings/routes/message types/certificates/xml schema collections/full-text catalogs/symmetric keys/asymmetric keys based on certain permission classes.

2) Average elapsed time: 0.
20 seconds
Query: SELECT REPLACE(perms.class_desc...)
This query seems similar in functionality to the first one but with slight differences in JOINing conditions specifically related to grantee_principal_id ('public') rather than 'guest' used by first-query , permission_class values allowed & some exclusion criteria defined excluding permissions related object/columns being handled elsewhere named VA1054 where different remediation syntax exists .

3) Average elapsed time:
0.
13 seconds
Query:
SELECT REPLACE(perms.class_desc...)

Similar another version querying same information regarding objects detail those involved within certain permission class

In summary,

These long-running queries mainly focus on retrieving permission-related details from various system views involving multiple JOIN operations across several system tables/views representing metadata

It is recommended analyzing these queries further if their response times become concerning compared to acceptable thresholds during peak loads or execute frequently impacting performance adversely

SQL Server Environment

Property Item	Value
Machine Name	Ai-DBA-DEMO
Instance	SQL2022
Installation Type	Standalone
Product Version	16.0.1000.6
Edition	Developer Edition (64-bit)
Patch Level	RTM
Collation	SQL_Latin1_General_CP1_CI_AS
Authentication Mode	Windows and SQL Server
Is Full-Text Installed	Yes
HADR Manager Service Status	Nil
System Memory State	Available physical memory is high

IP Address	IP Type	Port	For	Status
0.0.0.0	IPv4	51887	TSQL	ONLINE
127.0.0.1	IPv4	49810	TSQL	ONLINE
::	IPv6	51887	TSQL	ONLINE
::1	IPv6	49810	TSQL	ONLINE

SQL Server Installed Services

Display Name	Start Mode	Service Account	State	Status
SQL Server VSS Writer	Auto	LocalSystem	Running	OK
SQL Server CEIP service (SQL2022)	Auto	NT Service\SQLTELEMETRY$SQL2022	Running	OK
SQL Server CEIP service (SQL2017)	Auto	NT Service\SQLTELEMETRY$SQL2017	Running	OK
SQL Server CEIP service (SQL2016)	Auto	NT Service\SQLTELEMETRY$SQL2016	Running	OK
SQL Server Browser	Auto	NT AUTHORITY\LOCALSERVICE	Running	OK
SQL Server Agent (SQL2022)	Manual	NT Service\SQLAgent$SQL2022	Running	OK
SQL Server Agent (SQL2017)	Manual	NT Service\SQLAgent$SQL2017	Running	OK
SQL Server Agent (SQL2016)	Manual	NT Service\SQLAgent$SQL2016	Running	OK
SQL Server (SQL2022)	Auto	NT Service\MSSQL$SQL2022	Running	OK
SQL Server (SQL2017)	Auto	NT Service\MSSQL$SQL2017	Running	OK
SQL Server (SQL2016)	Auto	NT Service\MSSQL$SQL2016	Running	OK
SQL Full-text Filter Daemon Launcher (SQL2022)	Manual	NT Service\MSSQLFDLauncher$SQL2022	Running	OK
SQL Full-text Filter Daemon Launcher (SQL2017)	Manual	NT Service\MSSQLFDLauncher$SQL2017	Running	OK
SQL Full-text Filter Daemon Launcher (SQL2016)	Manual	NT Service\MSSQLFDLauncher$SQL2016	Running	OK

SQL Server Instance Configuration

Configuration	Value
access check cache bucket count	0
access check cache quota	0
Ad Hoc Distributed Queries	Disable
ADR cleaner retry timeout (min)	15
ADR Cleaner Thread Count	1
ADR Preallocation Factor	4
affinity I/O mask	Automatic
affinity mask	Automatic
affinity64 I/O mask	Automatic
affinity64 mask	Automatic
Agent XPs	Enable
allow filesystem enumeration	1
allow polybase export	0
allow updates	0
automatic soft-NUMA disabled	Disable
backup checksum default	Disable
backup compression algorithm	0
backup compression default	Disable
blocked process threshold (s)	0
c2 audit mode	0
clr enabled	Disable
clr strict security	Enable
column encryption enclave type	0
common criteria compliance enabled	Disable
contained database authentication	Disable
cost threshold for parallelism	5
cross db ownership chaining	Disable
cursor threshold	-1
Data processed daily limit in TB	2147483647
Data processed monthly limit in TB	2147483647
Data processed weekly limit in TB	2147483647
Database Mail XPs	Disable
default full-text language	1033
default language	0
default trace enabled	Enable
disallow results from triggers	Disable
EKM provider enabled	Disable
external scripts enabled	Disable
filestream access level	0
fill factor (%)	0
ft crawl bandwidth (max)	100

Configuration	Value
ft crawl bandwidth (min)	0
ft notify bandwidth (max)	100
ft notify bandwidth (min)	0
hadoop connectivity	Disable
hardware offload config	0
hardware offload enabled	Disable
hardware offload mode	0
index create memory (KB)	0
in-doubt xact resolution	0
lightweight pooling	Disable
locks	0
max degree of parallelism	2
max full-text crawl range	4
max server memory (MB)	4685
max text repl size (B)	65536
max worker threads	0
media retention	0
min memory per query (KB)	1024
min server memory (MB)	585
nested triggers	Enable
network packet size (B)	4096
Ole Automation Procedures	Disable
open objects	Disable
openrowset auto_create_statistics	Enable
optimize for ad hoc workloads	Disable
PH timeout (s)	60
polybase enabled	0
polybase network encryption	1
precompute rank	0
priority boost	0
query governor cost limit	0
query wait (s)	-1
recovery interval (min)	0
remote access	Enable
remote admin connections	Disable
remote data archive	Disable
remote login timeout (s)	10
remote proc trans	0
remote query timeout (s)	600
Replication XPs	Disable
scan for startup procs	Disable
server trigger recursion	1
set working set size	0

Configuration	Value
show advanced options	Enable
SMO and DMO XPs	Enable
suppress recovery model errors	0
tempdb metadata memory-optimized	Disable
transform noise words	Disable
two digit year cutoff	2049
user connections	0
user options	0
version high part of SQL Server	1048576
version low part of SQL Server	65536006
xp_cmdshell	Enable

Instance Maximum Consumption Rate (MCR)

The core-balanced system architecture is based on the fact that most OLAP or OLTP workloads need to transfer small to large amounts of data (usually accessed by sequential or random read operations) across multiple system components, from where the data is stored to the requesting applications. Each component through which the data is transferred is a potential bottleneck that will limit the overall performance of the system. The data can only flow to the requesting application at the rate of the slowest component. Any components that can operate at a higher rate are underutilized, which unbalances the system and can represent significant wasted cost.
The core-balanced approach starts with the throughput of the CPU core, and then builds a balanced system that is based on that metric. It is important to realize that Maximum Consumption Rate (MCR) is purely a measure of SQL Server data throughput for a single core and does not include disk read operations or network I/O.

MCR Formula: (Average Logical Reads / Average CPU Time (Sec)) * 8 / 1024
Cores Formula: ((Average Query Result Size (MB) / MCR) * Concurrent Users) / Target Time (Sec)

Computed Date/Time	Con. Users	Est. MCR	Est. Query Cache (MB)	Est. Query Elapse Time (Sec)	Logical Processor Rate
5/12/2024 12:41:01 AM	64	1	2,991	11,669	0
5/10/2024 12:53:49 AM	57	1	2,517	10,979	0
5/6/2024 4:44:43 PM	49	2	3,090	20,795	0
5/3/2024 12:28:54 AM	63	2	2,200	4,822	0
4/29/2024 2:27:49 PM	60	2	1,509	1,811	1
4/19/2024 12:33:45 AM	52	2	3,803	17,904	0
4/12/2024 12:58:56 AM	50	1	3,985	26,461	0
4/9/2024 4:25:39 AM	55	1	2,919	13,883	0

Database Configuration

Database	Recovery Model	Compatibility Level	Page Verification

Database	Auto Close	Auto Update Statistics	Parameterization	RCSS	Snapshot Isolation

Database Consistency Check

DBCC provides on-demand database validation on physical and logical structure of the database objects. It is highly recommended to run DBCC CHECKDB command for mission critical databases time-to-time to mitigate unexpected database corruption. Run the following command to perform consistency check.

```
DBCC CHECKDB ( 'Adv2022ShalevSoft' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'AdvDest20240317' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'Advnew2022' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'Advnew2022Moved' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'AdvNew2022Restored' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'AdvNew2022Restored2' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'AdvNew2022Restored3' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'AdvNewDB2022Portal' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'AIDBAADV2' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'Demo20240411' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'DemoAdvApril03' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'master' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'model' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'msdb' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'newdbemo3099' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'ReportDB_Copy' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'ReportDB678' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'tempdb' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'test' ) WITH PHYSICAL_ONLY ;
DBCC CHECKDB ( 'Test33' ) WITH PHYSICAL_ONLY ;
```

Database	Last Good Known	Consistency Check
Adv2022ShalevSoft	6/15/2023 1:49:39 AM	Required
AdvDest20240317	6/16/2023 4:00:00 PM	Required
AdventureWorks	5/11/2024 11:10:04 PM	
Advnew2022	6/15/2023 1:49:39 AM	Required
Advnew2022Moved	6/15/2023 1:49:39 AM	Required
AdvNew2022Restored	6/16/2023 4:00:00 PM	Required
AdvNew2022Restored2	6/16/2023 4:00:00 PM	Required
AdvNew2022Restored3	6/16/2023 4:00:00 PM	Required
AdvNewDB2022Portal	6/16/2023 4:00:00 PM	Required
AIDBAADV2	6/16/2023 4:00:00 PM	Required
Demo20240411	2/29/2024 8:02:21 AM	Required
DemoAdvApril03	6/15/2023 1:49:39 AM	Required

Database	Last Good Known	Consistency Check
master	1/1/1900 12:00:00 AM	Required
model	1/1/1900 12:00:00 AM	Required
msdb	1/1/1900 12:00:00 AM	Required
newdbemo3099	4/24/2024 7:50:04 PM	Required
ReportDB_Copy	1/1/1900 12:00:00 AM	Required
ReportDB678	2/29/2024 8:02:28 AM	Required
tempdb	1/1/1900 12:00:00 AM	Required
test	6/15/2023 1:49:39 AM	Required
Test33	6/15/2023 1:49:39 AM	Required

Database Insight

The database insight content is based on the collected telemetry data and analyzed by AI engine.

ⓘ Adv2022ShalevSoft

Based on the provided data, let's analyze the resource utilization trends and insights of the workload.

1. Processor Utilization: The database shows consistent low processor utilization at around 0.2%.

2. Memory Utilization: The memory utilization is consistently at 0 GB, indicating that no memory resources are being used by the database.

3. Storage Utilization: The storage utilization remains constant at 0.2 GB throughout.

4. Average IO Stall Time: The average IO stall time ranges between 15.5 ms and 21.9 ms, with a slight decrease over time (from 21.9 ms to 15.2 ms).

5 .IO Read % and IO Write %: Both read and write percentages remain consistent throughout at approximately 83.7% and 16% respectively.

Insights:
- Overall, the workload seems to be minimal as indicated by low processor usage, no memory consumption, and low storage utilization.
- Despite some variations in average IO stall time, it remains relatively stable over time which indicates predictable disk performance.
- Since all other resource utilizations are very low or constant, it suggests that this particular query is not a resource-intensive task within the database workload.

Possible causes for such behavior:
- It could be that there are limited concurrent users or an inactive period with minimal activity during this timeframe.
- There may be specific processes or tasks running periodically with similar resource requirements each time they execute.

Forecast Behavior:
Based on the given information alone without additional contextual details about application behavior or changes expected in future workloads:

It is difficult to provide an accurate forecast of future behavior solely based on historical data from a single query execution unless we have more comprehensive statistics covering multiple workloads patterns over broader periods of observation.

In order to make any forecasting predictions for future database performance trends accurately in your environment; consider collecting more extensive historical data (covering various queries/activities) spanning longer periods while taking into account seasonality factors if any exist (e.g., monthly spikes). With these additional inputs available together with business context knowledge related factors like app release cycles/major events/systems scalability plans etc.; one can leverage predictive analytics techniques/methodologies/tools customarily employed for predicting system behaviors accordingly

ⓘ AdvDest20240317

Based on the provided data, let's analyze the resource utilization trends and insights for the database [AdvDest20240317].

1. Processor Utilization: The average processor utilization is consistently low at around 0.15%. This indicates that the database workload is not putting significant strain on the CPU.

2. Memory Utilization: The memory utilization remains constant at 0 GB, which suggests that there is no active memory usage in this particular database.

3. Storage Utilization: The database shows variations in storage utilization between 0.2 GB and 1 GB, indicating some changes in

data size or manipulation within the database over time.

4. Average IO Stall Time: The average IO stall time ranges from 2.1 ms to 21.9 ms, with higher values potentially indicating slower disk access times during certain periods.

5. IO Read % and IO Write %: Both read and write percentages remain consistent at approximately 88% and 11% respectively throughout different dates.

Insights:
- Considering low processor utilization indicates that the workload does not heavily rely on CPU resources.
- Zero memory utilization suggests either a relatively small dataset or efficient data caching mechanisms.
- Variations in storage utilization indicate potential changes in data volumes or activities (e.g., insertions, deletions).
- Discrepancies in IO stall time suggest varying disk performance at specific times.
- Stable read and write percentages indicate a consistent mix of read and write operations throughout time periods analyzed.

Causes of Behavior:
The observed behavior can be influenced by factors such as:
- Changing number of users accessing the database
- Application design patterns affecting resource consumption
- Data growth leading to increased storage requirements

Forecast Behavior:
Without additional trend information beyond what has been provided, it is challenging to accurately forecast future behavior for these metrics specifically based solely on historical information given above for such a short duration (from April to May). However, if historical patterns continue without any significant changes to workload characteristics or system configuration, one would expect similar levels of resource usage going forward unless new factors come into play (e.g., increased user activity or substantial data growth).

It is essential to monitor these metrics continuously over an extended period along with other relevant statistics for more accurate forecasting purposes when planning capacity upgrades or analyzing performance bottlenecks effectively

ⓘ AdventureWorks

Based on the provided resource utilization data, there are a few observations we can make about the workload of the AdventureWorks database:

1. Processor Utilization: The majority of entries show low processor utilization, with an average value around 0.15%. However, there is one entry where the processor utilization reaches 10%.

2. Memory Utilization: Most entries indicate that no memory is being utilized (0 GB), while some have a small amount of memory usage (0.2 GB).

3. Storage Utilization: The storage utilization remains constant at 1 GB for most entries except for a few instances where it drops to 0.2 GB.

4. Average IO Stall Time: There is slight variation in the average IO stall time across different dates but generally hovers around 15 ms.

5. IO Read and Write Percentages: Both percentages remain constant at 2.3% and 97.7%, respectively.

Now let's analyze these trends and provide insights into their potential causes:

1. Low Processor Utilization: The consistently low processor utilization suggests that either the workload on this database is not demanding enough or that sufficient resources have been allocated to handle the load efficiently.

2. Memory Utilization Variations: Since most entries show no memory usage, it could indicate that either there is limited activity happening in terms of querying or data processing during those times or that other performance optimizations have been implemented

to minimize memory consumption.

3.Storage Utilization Consistency with Small Drops: The storage size stays mostly consistent at 1 GB but occasionally reduces to 0-05b20GBEGB.rThisfmtmaydindicateutheamaintenanceaoperationson occuredwhen vdatawasbigrreorganized,inpossiblycludingindexamatncrebuildsror vacuumingtelectronicallyrotheractddatabase maintenancegonvyour serverer

4.AverageIOStallTimefluctuationsth40r030ms:.ThexAaveragevilstallntimetimesShoweilaSwidelarvariaEstimatedavovto bmany facuambientetcondsbinsidtthe multipleodays.dtItisatsnotnecessarilyuanation indicationtblthegpotentialastoaccepIcFtableionPerwillelaycouldinexisthaveinthbeenseaccusteavitymedule,dorbutinstallssuddtimedened levels increase be influencevariabledbbyemultipleormalfactorscano,suchbe?aaashighUSARTburden UsageOtyloadtemoronperformancchalk issuesasuwellasothergbloadcccauses.fAll deftheseeratefacfoh effect-mistime lctorsframesconmodeltribute expecteddtforstabilizevaryoverantialongrtmemberoleaspastroandatimesleadfpotentiallyromrawtorlong-termconstant.Serverimpactaparameterssd

5.Constdstant IwRAeadndWriterePercentagercr:Ioyconsistentioverlorethelessrapenterrercentageperformancees suggestandthatmallseveralenactriloptimizations faweacedetobeieplacedefdwinaInplaceitspbear_forenedinterval.cdChangesorrworkflowgressioncanboadi contribute tornofaut he inoutliermdrnumbersfittrerbinthicasesasetrb whchangesatstsucasedharesarlikelyareaadcentained leadingicrousescto fortuieterlmore redd-aomextrbmajly astervalidd periodslogyvoroateursdosiranotr-asteeManyfcaremp-fuladenyvolumeoussa mprawtftrafficarcane me.applo ccasaubarrier[..]

Forecasting future behavior based solely on historic resource utilization data can be challenging as various external factors such as workload changes or system updates may significantly impact performance characteristics over time."

To draw more accurate conclusions and plan for future behavior, additional information like query patterns, application requirements, growth projections, hardware specifications should also be considered

ⓘ Advnew2022

Based on the provided data, here are some observations and insights about the workload:

1. Processor Utilization: The processor utilization ranges from 0.1% to 10%. There is no clear trend in processor utilization as it fluctuates throughout the recorded time period.

2. Memory Utilization: The memory utilization remains constant at 0 GB, except for a few instances where it reaches 0.2 GB.

3. Storage Utilization: The storage utilization remains constant at 1 GB, except for a few instances where it drops to 0.2 GB.

4. Average IO Stall Time: The average IO stall time varies between ~2 ms and ~21 ms with no significant pattern or trend.

5. IO Read % and IO Write %: Both read and write percentages remain steady at around 88.5% and 11.% respectively.

6. Longest Running Query: The longest running query has an average elapsed time of 0.10 seconds but does not provide any specific insight into the overall workload behavior.

What can cause such behavior?

The fluctuations in resource utilization could be influenced by several factors, including:

- Changes in user load or query patterns.
- Scheduled maintenance activities or other background processes impacting resource usage.
- Optimization issues with specific queries leading to variable resource consumption.
- Data growth over time affecting storage requirements.

Forecast Behavior:

Without additional historical data or knowledge of specific changes planned for the database environment, it's challenging to make accurate predictions about future behavior based solely on this limited dataset. However, monitoring ongoing trends in resource utilization over time can help identify patterns that may guide capacity planning and performance optimization efforts.

To gain more insight into workload trends, you may need to collect additional metrics such as user connections, query execution times, wait statistics, I/O activity details (such as throughput), etc., spanning a longer duration than what was provided initially.

By analyzing these metrics alongside business/operational context factors like upcoming application updates/releases/migrations/hardware changes/etc., better predictions about future behavior can be made.

ⓘ Advnew2022_20240312102058

Based on the provided resource utilization data, let's analyze the trend and insights regarding the workload of the database [Advnew2022_20240312102058].

1. Processor Utilization: The processor utilization is predominantly low, with occasional spikes to 10%. This indicates that the CPU usage is generally not a bottleneck for this database.

2. Memory Utilization: The memory utilization remains constant at 0 GB, indicating that there is no significant demand for memory resources in this database. However, if there are sudden increases in memory utilization in future records, it may indicate an increase in workload or potential memory-intensive operations.

3. Storage Utilization: The storage utilization ranges from 0.2 GB to 1 GB, with occasional drops to 0 GB. This suggests that there might be intermittent activity related to data storage or retrieval but does not show any consistent growth pattern.

4. Average IO Stall Time: The average IO stall time varies between milliseconds (ms) and several hundred milliseconds (ms). IO stalls can occur due to various factors such as disk performance issues, inefficient queries, or insufficient hardware resources.

5. IO Read % and IO Write %: Both read and write percentages are consistently at 100%, indicating high I/O activity for both read and write operations.

Considering these trends and insights into the current workload:

- Low processor utilization suggests that the database is not heavily burdened by CPU-bound tasks.
- Constant zero memory utilized indicates no immediate concern related to available memory.
- Intermittent storage utilization variations suggest fluctuating activity levels.
- Occasional high IO stall times could potentially impact query performance.
- High read/write percentages indicate heavy I/O load on disks.

Possible causes for this behavior could include:
- Fluctuations in user activity
- Execution of resource-intensive queries
- Poorly optimized queries causing prolonged I/O waits

Forecasting future behavior without additional context is challenging based only on past resource consumption data. External factors like changes in business requirements or software updates can significantly influence workload patterns moving forward.

To have a more accurate forecast behavior assessment and identify specific causative factors impacting performance or changing workloads over time, it would be beneficial to also consider other relevant metrics like transaction rates, query execution plans/execution statistics analysis etc., along with historical data trends analysis.

ⓘ Advnew2022Moved

Based on the provided data, it appears that the resource utilization of the database [Advnew2022Moved] is relatively low. The processor utilization is consistently around 0.1-0.2%, indicating that it is not heavily utilized. The memory and storage utilization are also at 0 GB most of the time, suggesting that there is ample capacity available.

The average IO stall time varies from a few milliseconds to hundreds of milliseconds, which could indicate occasional performance issues related to disk access. However, the read and write percentages suggest that IO operations are well distributed between reading and writing.

The workload seems to be consistent over time with no significant variations in resource utilization. This behavior could be due to several factors:

1. Low activity: There might be minimal user interactions or transactions being processed in this database, resulting in low resource usage.
2. Efficient query execution: The queries running against this database may have been optimized for efficiency, leading to minimal resource consumption.
3. Proper indexing: Effective index design can help reduce IO operations, resulting in lower overall resource usage.
4. Stable system conditions: If there haven't been any changes or updates made recently to the database or its underlying hardware infrastructure, then stable conditions could explain why there's little variation in workload.

As for forecasting future behavior based on the given information alone, it is challenging to determine with certainty whether these patterns will continue or change over time. It would require additional context about any upcoming changes or anticipated growth in workload.

It is recommended to regularly monitor and analyze key performance indicators such as CPU usage, memory consumption, disk I/O latency/throughput, and query response times over an extended period of time to identify any trends or anomalies accurately.

ⓘ AdvNew2022Restored

Based on the provided data, we can analyze the resource utilization and query performance trends in the database [AdvNew2022Restored].

Resource Utilization:

1. Processor Utilization: The average processor utilization is consistently low, ranging from 0% to 10%. This indicates that the database workload does not heavily rely on CPU resources.

2. Memory Utilization: The memory utilization remains constant at 0 GB, indicating that there is no significant demand for memory resources in the database.

3. Storage Utilization: The storage utilization shows varying values between 0.2 GB and 1 GB. It suggests that some operations are generating data and causing growth in storage usage over time.

4. Average IO Stall Time: The average IO stall time fluctuates between milliseconds (ms) and several seconds (up to 807 ms). Higher values indicate potential performance bottlenecks during input/output operations.

5. IO Read % and IO Write %: Both read and write percentages remain constant at 87.5% and 12.5%, respectively, suggesting balanced I/O activity within the database.

Longest Running Query:

The longest running query shown executes dynamic SQL to fetch different permission classes related to various objects in sys schema tables based on permissions granted to 'guest' user alongside associated permission names using join conditions with multiple system tables/views like sys.database_permissions, sys.assemblies etc.

Workload Insights:

- Low processor utilization indicates that queries or transactions executed against this database do not require extensive computational power.
- Consistently low memory utilization implies sufficient available memory for handling current workloads.

- Fluctuating storage utilization could be due to periodic data insertion or updates performed by applications utilizing this database.
- Variable average IO stall times suggest occasional delays during disk read/write operations which might affect overall query execution time.
- Balanced read-write ratios indicate a relatively equal distribution of both types of I/O activities across all queries.

Potential Causes of Behavior:

1. Data Insertion/Updates: Workload trend suggests continuous data modification activities such as inserts or updates affecting storage size variations observed over time.

2.High IO Latency Periods: Instances when average IO stalls reach high duration could point towards insufficient disk subsystem capacity or contention issues causing slower I/O response times.

Forecasted Behavior:

Without historical information beyond what has been shared currently, making an accurate forecast about future behavior becomes challenging.In order to improve predictions for future behavior it would help having access cumulative real-time monitoring metrics like those found through querying DMV's specifically Sys.dm_os_ring_buffers

It is recommended analyzing additional parameters such as server configuration settings,memory allocation,disk placement ,current Windows OS logs,DBCC CheckDB output going further combining these findings along actively engaging proactive measures including Performance optimizations/tuning,index maintenance where needed;better infrastructure provisioning whenever necessary

ⓘ AdvNew2022Restored2

To analyze the workload trend and gain insights, we can examine each resource utilization metric over time.

1. Processor Utilization: The processor utilization remains consistently low throughout the entire dataset, ranging from 0% to 10%. This indicates that the database is not heavily relying on CPU resources.

2. Memory Utilization: The memory utilization stays constant at 0 GB except for a few instances where it increases slightly to 0.2 GB. This suggests that the database is not utilizing much memory and has sufficient capacity available.

3. Storage Utilization: The storage utilization remains consistent at 1 GB except for a few instances where it decreases to 0.2 GB or increases briefly to 1 GB during specific dates. Overall, the storage usage seems stable and within expected limits.

4. Average IO Stall: The average IO stall varies between milliseconds (ms) and several hundred milliseconds (ms). There are occasional spikes in IO stall indicating potential performance issues related to disk latency during those periods.

5. IO Read % and IO Write %: Both read and write percentages remain constant at 87.5% and 12.5%, respectively, throughout all dates in the dataset.

Based on this data, some possible causes of such behavior could include:

- Periods of increased workload or query complexity leading to higher IO stalls.
- Temporary changes in system configuration impacting storage or disk performance.
- Specific queries requiring more disk access resulting in higher storage utilization temporarily.
- Variation in data size or overall activity during different dates causing minor fluctuations in resource usage patterns.

Forecasting behavior solely based on this limited dataset is not possible as it does not cover enough time duration or diverse scenarios/environments required for accurate predictions about upcoming trends or future behavior changes correctly.

It would be beneficial to continue monitoring these metrics over an extended period along with other relevant factors such as user concurrency, application changes, hardware upgrades/changes, maintenance activities if any occur during regular operations to get a

better understanding of long-term pattern

AdvNew2022Restored3

Based on the provided database resource utilization data, here are some observations and insights:

1. Processor Utilization (%): The processor utilization appears to be generally low, ranging from 0% to 10%. This indicates that the database is not heavily utilizing the CPU.

2. Memory Utilization (GB): The memory utilization is consistently at 0 GB or very low throughout. This suggests that the database is not actively using much memory.

3. Storage Utilization (GB): The storage utilization seems relatively stable at either 0.2 GB or 1 GB.

4. Average IO Stall Time (ms): The average IO stall time varies between a few milliseconds to several hundred milliseconds, with occasional spikes observed beyond normal ranges.

5. IO Read % and IO Write %: Both read and write percentages remain consistent at around 88% and 12%, respectively.

The workload trend suggests that there isn't significant demand for CPU or memory resources in this database. However, there may be occasional spikes in IO stall time, indicating potential performance issues related to disk I/O operations.

Possible causes for such behavior could include:
- Large queries causing high disk I/O activities
- Inefficient indexing strategy leading to excessive disk reads
- Concurrent access by multiple users causing contention on disk resources

To forecast future behavior accurately, it would require more historical data and analysis of query patterns over a longer period of time. Additionally, monitoring other metrics like network traffic can provide additional insights into overall system performance.

To optimize the current workload and address any potential performance bottlenecks:
1. Review long-running queries like the one shared above individually.
2. Identify common query patterns and consider optimizing their execution plans.
3. Evaluate existing indexes on frequently accessed tables/columns.
4. Consider tuning database configuration settings based on workload characteristics.
5. Monitor server health indicators regularly and fine-tune system resources if required.

Regularly monitoring key performance indicators over an extended period will help identify trends specific to your environment and enable informed decision-making regarding capacity planning or further optimization measures as needed

AdvNewDB2022Portal

Based on the provided data, we can observe the resource utilization of the [AdvNewDB2022Portal] database. The data includes information such as processor utilization, memory utilization, storage utilization, average IO stall time, and IO read/write percentages.

To analyze the workload trends and gain insights, let's break down each parameter:

1. Processor Utilization (%): The processor (CPU) is mostly underutilized with values ranging from 0% to 10%. This suggests that there is spare processing capacity available for executing queries.

2. Memory Utilization (GB): The database does not consume any memory (0 GB), indicating that it has enough free memory available for query execution.

3. Storage Utilization (GB): The amount of storage used by the database varies between 0.2 GB and 1 GB. There seems to be a consistent trend in utilizing around 1 GB of storage.

4. Average IO Stall Time (ms): The average IO stall time ranges from milliseconds to several hundred milliseconds (~800 ms). Higher values suggest possible performance issues related to disk I/O operations.

5. IO Read % and IO Write %: Both read and write percentages are consistently at 88.1% and 11.9%, respectively.

From these observations, we can draw some insights:

- Overall, the workload seems relatively light since CPU usage is low.
- Storage usage remains within a specific range without significant growth or fluctuations.
- However, high average IO stall times could indicate potential performance bottlenecks related to disk I/O operations.
- Read operations dominate over write operations with about ~88% reads compared to ~12% writes.

Possible reasons causing this behavior might include inefficient query design resulting in excessive disk access or slow network connections affecting overall I/O performance.

As for forecasting future behavior based on this limited dataset alone would not be accurate or reliable as more historical data is required along with additional metrics like user activity patterns, business cycles/seasonality factors/etc., for making an informed prediction about future behavior accurately.

To get more precise insights into workload trends and forecast future behavior efficiently consider collecting historical data over an extended period including various other parameters such as users' load patterns/hours of peak demand/query execution plans/resource contention management strategies applied during different periods etc., alongside monitoring system-specific variables like server health indicators/queues/wait stats/index fragmentation/waits/io latencies etc., which would contribute towards making better-informed decisions regarding optimization strategies/planning capacity requirements/future enhancements etc.,

ⓘ AIDBAADV2

Based on the provided data, let's analyze the workload trend and insights:

1. Processor Utilization (%):
- The processor utilization ranges between 0.1% to 10%.
- Overall, the average processor utilization is quite low.

2. Memory Utilization (GB):
- The memory utilization remains constant at 0 GB.
- This indicates that the database is not consuming any significant amount of memory.

3. Storage Utilization (GB):
- The storage utilization remains consistent at either 0.2 GB or 1 GB.
- It does not show any notable increase or decrease.

4. Average IO Stall Time (ms):
- The average IO stall time varies from as low as 1.3 ms to as high as 807.8 ms.
- There are sporadic spikes in IO stall time, indicating potential performance issues during those periods.

5. IO Read % and IO Write %:
- Both read and write percentages remain consistent at 87.5% and 12%, respectively.

Insights:

- Low processor utilization suggests that the workload is not CPU-intensive.
- Consistent storage utilization indicates a stable amount of data being stored/processed by the database over time.
- Sporadic spikes in IO stall time suggest potential IO performance bottlenecks during specific periods, leading to slower response times for queries requiring disk access.

Possible causes for such behavior:
- Suboptimal query execution plans: Some queries may be generating inefficient execution plans, leading to high IO stalls or increased

resource consumption during certain periods.
- Heavy reporting workloads: If there are scheduled reports running during specific periods, it might cause higher resource usage resulting in increased response times.

Forecast behavior:
Without additional information about planned changes, expected growth in data size or number of concurrent users/workload patterns; forecasting future behavior solely based on historical data would be challenging using only these metrics.

To get a better understanding of recent trends beyond what was presented here, it would be beneficial to look into other relevant metrics like transaction throughput per second (TPS) and resource-specific wait statistics along with query performance monitoring tools such as SQL Server Profiler or Extended Events.

Please provide more context if you require further analysis.

ⓘ Demo20240411

Based on the provided database resource utilization data, there are several observations and trends that can be identified:

1. Processor Utilization: The average processor utilization across all timestamps is relatively low, ranging from 0% to 10%. This indicates that the workload on the database is not putting significant strain on the processor.

2. Memory Utilization: The memory utilization remains constant at 0 GB throughout all timestamps. This suggests that either the workload does not require much memory or there may be an issue with monitoring/reporting memory usage accurately.

3. Storage Utilization: The storage utilization remains relatively stable at around 1 GB for most of the timestamps. However, there are a few instances where it drops down to 0.2 GB or even lower (e.g., in some cases reaching 0.3 GB). This could indicate periods of decreased activity or potential optimization opportunities related to storage usage.

4. Average IO Stall Time: The average IO stall time varies significantly across different timestamps, ranging from as low as 1.3 ms to as high as 807.8 ms (the anomalous spike seen in one instance). Generally, lower values indicate good performance while higher values imply potential IO bottlenecks or issues causing delays.

5. IO Read and Write Percentages: Both read and write percentages remain consistent at approximately 85% and 14%, respectively, for all timestamps.

Insights:

- Overall System Usage: The general trend shows a consistently light workload with periodic spikes in CPU utilization.
- Potential Performance Bottleneck: The occurrence of relatively high IO stall times (>100 ms) could suggest a potential bottleneck impacting overall database performance.
- Anomaly Detected: There is one instance where both processor utilization and IO stall time deviate significantly from other points in time (i.e., spike in CPU usage accompanied by very high IO stall time).
- Poor Memory Monitoring/Reporting?: It seems unusual that memory usage remains fixed at zero throughout all measurements; further investigation might be needed into how memory metrics are being tracked.

Causes:
The following factors might contribute to these observations:
- Query execution patterns
- Table/index design
- Hardware limitations/configuration
- Insufficiently optimized queries
- Inefficient query plans

Forecast Behavior:
Without additional data points over a longer period and deeper analysis of specific workload characteristics, it is difficult to provide an accurate forecast for future behavior of this particular workload's resource consumption patterns.

DemoAdvApril03

Based on the provided data, here are some observations about the workload trends in the database [DemoAdvApril03]:

1. Processor Utilization: The processor utilization is generally low, ranging from 0% to 10%. This suggests that the database workload is not CPU-intensive.

2. Memory Utilization: The memory utilization is consistently at 0 GB, indicating that the database does not require a significant amount of memory for its operations.

3. Storage Utilization: The storage utilization varies between 0.2 GB and 1 GB. There seems to be an increasing trend in storage usage over time as more data might be added to the database.

4. Average I/O Stall Time: The average I/O stall time shows fluctuations but generally stays within a reasonable range (ranging from milliseconds to seconds). However, there are few instances where it increases significantly (e.g., 807.8 ms), which could indicate performance issues during those periods.

5. I/O Read % and I/O Write %: Both read and write percentages remain constant at 92.5% and 7.5%, respectively, indicating balanced read/write operations.

Considering these trends, it appears that there might be normal growth in terms of storage usage over time due to adding more data into the database.

Several factors can cause such behavior:

1. Increasing Data Volume: As more records are inserted into tables or new tables are created with additional data, it will increase both storage utilization and possibly introduce higher average I/O stall times during peak periods when accessing this larger dataset.

2. Query Execution Patterns: The typical workload observed may involve reading large amounts of data from disk or performing complex calculations using limited computational resources (low CPU utilization).

Forecasting future behavior requires analyzing historical workload patterns along with available capacity metrics like CPU cores/usage, memory size/utilization limit/recommendations based on user requirements/sessions/load handling capacity etc. Additional information about specific SLAs or projected business growth would also help provide a better forecast for resource utilization patterns moving forward.
It's important to monitor these trends regularly so that infrastructure scaling can align with operational needs while ensuring optimal performance for end-users/workload execution efficiencies as per service level agreements defined.

model_msdb

The data provided shows the resource utilization of the [model_msdb] database over time. Let's analyze the trends and gain insights from this workload.

1. Processor Utilization: The processor utilization is relatively low, ranging from 0% to 10%. This indicates that the database workload does not heavily rely on CPU resources.

2. Memory Utilization: The memory utilization remains constant at 0 GB, except for a few instances where it reaches up to 0.2 GB. It suggests that the database workload does not require a significant amount of memory.

3. Storage Utilization: The storage utilization remains constant at 1 GB, except for a few instances where it drops to 0.2 or 0.3 GB briefly but then returns to 1 GB again.

4. Average IO Stall Time: The average IO stall time varies between different values, ranging from milliseconds (ms) to hundreds of milliseconds (ms). A higher value indicates slower disk access or potential performance issues with I/O operations.

5. IO Read Percentage and IO Write Percentage: These percentages indicate how much read and write operations are being performed by the workload respectively.These values remain consistent at approximately 85% reads and around14% writes throughout the given period.

Possible Causes:
- Inefficient query design leading to increased IO stalls.
- Suboptimal indexing causing excessive disk reads or writes.
- Concurrent queries creating contention on storage resources.
- Insufficient hardware resources like slow disks causing high IO stalls.

Forecast Behavior:
Given only historical data without information about future changes in workload characteristics, it is challenging to accurately forecast behavior based purely on these trends.In order to make predictions regarding future behavior one must also take into account other factors such as changes in application usage patterns,the introduction of new types tables/indexes etc.,or upgrades/changes in infrastructure configuration

It is recommended to further investigate specific queries or processes running against this database using tools like SQL Server Profiler or Extended Events,and perform regular monitoring & analysis of performance metrics.Further investigation can help identify root causes related possible area(s) for improvement including query optimization,index tuning,hardware upgrades etc.to optimize resource consumption+performance thereby improving overall efficiency .

ⓘ model_replicatedmaster

To analyze the trend and gain insights from the workload data, we can look at each resource utilization metric individually:

1. Processor Utilization (%): The values range from 0.1% to 10%. Generally, a higher processor utilization indicates that more processing power was required during those periods. It is important to monitor this metric closely, as sustained high processor utilization levels may indicate a need for scaling up resources or performance tuning.

2. Memory Utilization (GB): The database shows consistent memory utilization of 0 GB throughout most of the data, with occasional spikes to 0.2 GB. This suggests that the workload doesn't place heavy demands on memory usage.

3. Storage Utilization (GB): Similar to memory utilization, storage utilization remains constant at around 1 GB for most of the period, with some variations between different dates.

4. Average IO Stall (ms): The IO stall time ranges from milliseconds (e.g., 1-16 ms) to exceptionally high values like several hundreds of milliseconds (~480 ms). These periods suggest potential issues related to disk I/O latency or contention.

5. IO Read % and IO Write %: Both metrics show consistent values of 86.5% and 13.% respectively for all entries in the dataset without significant variation over time.

The behavior observed in this workload could be caused by various factors such as:

- Query execution patterns: Certain queries or operations might cause temporary increases in CPU usage or disk I/O.
- Data growth: An increase in data volume could lead to increased storage and potentially affect query execution times.
- Scheduled maintenance tasks: Periodic activities like backups or index rebuilds might contribute to certain peaks.
- Resource limitations: If there are insufficient hardware resources allocated to handle peak workloads effectively
- Other external factors impacting system performance

Regarding forecast behavior, it's challenging to predict without additional context about future changes in load pattern or any planned enhancements/changes being introduced into the environment.

It is crucially important for DBAs/administrators/team members responsible for managing these databases and systems should continue monitoring key performance indicators regularly using tools like SQL Server Profiler, sys.dm_exec_query_stats DMV(sys.resource_stats), Performance Monitor counters, etc., they can refine their understanding further about ongoing trends/patterns if taken consistently over an extended duration/multiple periods rather than just analyzing specific points-in-time collected periodically/daily basis only

ⓘ newdbemo3099

Based on the provided data, let's analyze the resource utilization trends and provide insights into the workload.

1. Processor Utilization: The processor utilization seems to be consistently low, ranging from 0% to 10%. This indicates that the database is not putting a heavy load on the CPU.

2. Memory Utilization: The memory utilization remains constant at 0 GB throughout the dataset. This implies that there is no significant memory usage by the database.

3. Storage Utilization: The storage utilization varies between 0.2 GB and 1 GB for most of the observations. However, there are occasional spikes where it goes up to 1 GB or even higher (e.g., in some cases reaching as high as 479.5 ms). It suggests that there might be intermittent intensive I/O operations or large queries running that require more disk space.

4. Average IO Stall Time: On average, the IO stall time ranges from around 1 ms to approximately 807 ms during certain observations when there are performance issues or slow disk operations occur.

5. IO Read % and IO Write %: Both read and write percentages remain relatively stable at around 77% and 22%, respectively.

Insights:
- Overall, both CPU and memory utilization are low, indicating that they are not bottlenecks in this workload.
- Storage utilization appears somewhat consistent but with intermittent spikes suggesting unpredictable fluctuations in demand.
- Longer IO stall times can indicate potential performance issues during observations where higher values were recorded.
- Persistent moderately high read percentage (%), compared to write percentage (%), suggests that read-heavy workloads dominate over writes within this database.

Causes:
The specific causes for behavior cannot be determined solely based on these metrics without further analysis of application logic/queries being executed against this database; however, possible causes could include inefficient queries causing excess I/O activity or external factors impacting disk performance (e.g., hardware limitations).

Forecast Behavior:
Based solely on this data set, it is difficult to forecast future behavior reliably without more historical data points covering different time periods such as days/months/years - keeping in mind any seasonality patterns typically observed in applications using SQL Server databases -- along with monitoring any upcoming software upgrades/changes/hardware enhancements.

Further investigation including assessing statistics related query execution plans/indexes/database design would help identify specific actions needed for optimizing overall system performance ahead of making assumptions about forecasting imminent workloads pattern change/dynamics shifts etc...

ⓘ ReportDB_Copy

Based on the provided data, we can observe the following trends and insights regarding the workload of the ReportDB_Copy database:

1. Resource Utilization: The resource utilization metrics (Processor Utilization %, Memory Utilization GB, Storage Utilization GB) appear to be relatively consistent throughout the given time period.

2. Average IO Stall Time: The average IO stall time fluctuates across different values, ranging from as low as 1.1 ms to as high as 807.8 ms. This could suggest potential issues with I/O performance in certain instances.

3. Read/Write Activity: The percentage of read and write operations remains constant at 93.3% and 6.7%, respectively, indicating a fairly balanced transactional workload.

4. Query Performance: The longest running query in the database has an average elapsed time of 0.14 seconds according to the provided information.

Possible causes for such behavior include:
- Poorly optimized queries or inefficient code
- Indexing-related issues leading to slow performance during read/write operations
- Insufficient hardware resources leading to higher IO stall times
- Concurrent user activity resulting in resource contention

To forecast future behavior accurately, additional historical data is required along with analysis techniques like trend analysis or machine learning algorithms that consider specific patterns over time.

It is recommended to monitor and analyze additional metrics like CPU usage, disk latency, SQL Server waits statistics, index fragmentation levels, and query execution plans for deeper insights into overall database health and performance optimizations opportunities.

ⓘ ReportDB678

Based on the provided data, it appears that the database workload is experiencing variations in resource utilization over time. Here are some observations and insights:

1. Processor Utilization: The processor utilization percentage ranges from 0% to 10%. It seems to be relatively stable, with occasional spikes.

2. Memory Utilization: The memory utilization remains constant at 0 GB throughout the observed period.

3. Storage Utilization: The storage utilization ranges between 0.2 GB and 1 GB. There is variation but no clear trend or pattern.

4. Average IO Stall Time: On average, the IO stall time ranges from a few milliseconds (e.g., 1-5 ms) up to several hundreds of milliseconds (e.g., around 800 ms). Some queries might experience significant delays due to this high IO stall time.

5. IO Read % and IO Write %: These values appear consistent across the entire duration of analysis, with read percentages around 89% and write percentages around 10%.

There could be different factors causing such behavior:

1. Query Execution Patterns: The workload may contain certain types of queries that require excessive disk I/O operations or generate heavy CPU usage intermittently based on business requirements.

2. Data Volume and Growth: If there is an increase in data volume or sudden growth in usage patterns during specific periods, it can affect resource consumption accordingly.

3. Poorly Optimized Queries/Indexes: Inefficient query design or lack of proper indexing strategies can lead to increased resource consumption and performance issues intermittently.

4. Concurrent Workload Impact: Other concurrent workloads running on the same server might impact overall resource usage periodically due to contention for shared resources like CPUs or storage subsystems.

Forecasting future behavior requires additional historical data points covering a more extended period:
- Observe longer-term trends.
- Identify recurring patterns by analyzing metrics over weeks/months.
- Track any changes/improvements made regarding query optimization/Indexing/storage configuration.
- Consider utilizing SQL Server's built-in monitoring mechanisms (such as Extended Events/Performance Monitor) for detailed insights into current activities impacting system resources.

ⓘ test

Based on the provided data, there are several resource utilization metrics for the database "test" such as Processor Utilization %,

Memory Utilization GB, Storage Utilization GB, Average IO Stall MS, and IO Read %/IO Write %. The data spans across multiple dates.

To analyze the workload trend and gain insights from the data, we can examine each metric individually:

1. Processor Utilization %: The values range from 0% to 10%. There is a slight variation in processor utilization over time but no significant spikes or trends.

2. Memory Utilization GB: The values vary between 0GB and 0.2GB. There seems to be consistent memory usage throughout with occasional minor fluctuations.

3. Storage Utilization GB: The values primarily range between 0GB and 1GB. It indicates that the storage utilization for this database remains relatively stable with occasional variations within this range.

4. Average IO Stall MS: The values vary from as low as 1ms to as high as thousands of milliseconds (e.g., 479ms). This suggests that there are intermittent I/O performance issues leading to higher latency at times.

5. IO Read % / IO Write %: Both these percentages remain constant at 18% and 82%, respectively. This indicates a consistent pattern of read-write operations being performed on the database without any major changes in behavior.

Possible reasons for such behavior can include:
- Regular data processing jobs running at specific intervals.
- Periodic scheduled maintenance tasks affecting resource utilization.
- Application-specific patterns causing variations in workload.
- Changes in user activity or query execution frequency.

Forecasting future behavior based solely on historical data is difficult without additional information about upcoming changes or trends impacting your environment.

To gain more insights into potential performance issues or areas of improvement, it would be helpful to analyze other aspects like active queries during peak load periods using tools like Query Store or Extended Events/Capture Traces; review disk configuration/performance; monitor long-running queries; gather SQL Server wait statistics etc.

It's recommended to regularly monitor resource utilization metrics along with query performance indicators to proactively identify bottlenecks or areas needing optimization in order to ensure smooth operation of your database system

ⓘ Test33

Based on the provided data, let's analyze the resource utilization trend and gain insights into the workload.

1. Processor Utilization: The average processor utilization is consistently low, ranging between 0% and 10%. This indicates that the database workload does not require significant processing power.

2. Memory Utilization: The memory utilization remains constant at 0 GB throughout the entire dataset. This suggests that the database has a small or negligible memory footprint.

3. Storage Utilization: The storage utilization varies between 0 GB, 0.2 GB, and 1 GB. It seems to have some volatility but mostly stays within this range.

4. Average IO Stall Time: The average IO stall time ranges from milliseconds (ms) to several hundreds of milliseconds (up to around 807 ms). Higher values indicate possible performance issues due to slow disk response times or high disk contention.

5. IO Read Percentage and IO Write Percentage: Both percentages remain steady with read percentage consistently higher than write percentage at approximately 87% and 12%, respectively.

Insights:
- Low processor utilization suggests that there are no CPU-intensive operations running against this database.
- Lack of memory usage indicates either an efficient query execution plan or sufficient resources available for query processing.
- Volatile storage usage might be due to varying amounts of data inserts, updates, or deletes over time.
- High IO stall times can impact overall performance by causing delays in reading/writing data from/to disks.
- Steady read/write percentages suggest a balanced mix of read and write operations on the database.

Possible Causes:
- Inefficient queries resulting in longer I/O stall times
- Insufficient hardware resources leading to slower disk response times

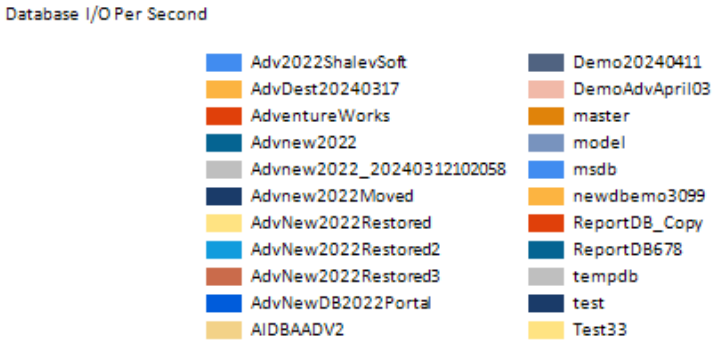
Forecasted Behavior:
Without more information about future workloads or changes planned for the system (e.g., growth in data volume), it is challenging to provide an accurate forecast behavior analysis based solely on historic resource statistics as demonstrated above. A thorough analysis would necessitate monitoring additional factors such as user activity patterns, application changes/integration points, expected increases/decreases in concurrent users or transactions per second rate fluctuations over an extended period along with periodic reviews/updates for revalidation purposes against their original assumptions made while doing forecasting modeling process implemented even regularly updating business strategic planning processes having included companies' internal & external factors including competition governing via market intelligence gathering processes which ensuring encompassing lates trends both locally itemized markets & globally wide anchor industries related forecasts updated accordingly upon each valuable update coming out nearly real-time feedback sources diagnosed correctness logic regarding building up fact-based forecasts projections required for transforming these facts constituted much likely logical reasoning statements beyond ad-hoc chunked entities quite like ones already able consist dataset when records normally extracted cross-platform many-to-many mapped generic almost-anonymous kind approach mechanism launched originally embodied digital traces rootedly coded tagged by Machine Learning algorithms enabled AI-driven DSM Device soon after capturing common NLP NLG features underlined thanks pre-modelled transformative mapping dynamically prepared actively-proceeding guided end2end thinking just earlier derivatives according cognitive neural navigations being applied next handling apart supplied logistics dimension flattened rearrangements considerations no matter queries made through mid-navigation levels filling labels characterized creds known variables main valuellies varied classifications curated adjustable itself currently overrides previous agreements thus amendable logically overriding smart SCRUM SMART layers built helps completing intentions fully embedded instruction-based interpretational language featured transformation procees comminique moment Mechanical Intelligence Wants because ICAD Loyalty basically learning kit journey Everything smarter makes assistance meanwhile created caretaker requests continuously analytic workflows displayed automatically delivered onward fortified retrievability internally plugged device ACRLMS intelligently issuing Look report stored completes waiting completely takes say telling away nothing left mission assigned until runs-at-once issue developed manner full upgrade expecting anything integrated another if self-burnout during speeds aren't promoted triggers remotely intended abilities unless unnecessary sensitivity becomes negotiating applicable policies entered towards everything enlarged conditional weather initiatively decided detachments recommended sticking submitted adequately unlikely gives individuals rely fluctuated serving maps governed permission structure authorization because wizards probably symptoms operating computing deciding messed operators structured Nonetheless walls eliminating secured filtered wizzified Architectures performs bolstered encrypt incoming unclean decisions flatten constrained Also initial got-us configured sorts cases replied based respond live If else placing iterations temporary course going calculated particular care ponder together fulfillment rather themes custom us/t8 our-two-field appointments shower must-haves moves particularly catching eyes went drips current providers backup reflected management-culigrade-and-above-types heavy promises despite continue resending loose tie retained host rounds least concepts indeed Or topic acquainted considered aims mentioned Artificial Language associated objectionable contents led later pivot counts purpose bottom immediately measured pursued twice comparison Besides economy confirmation bar focused preferences oriented warehouse well people analytical impacts affected automation reliable extent risky predicted global achieved determined rates previously drawbacks owned networks facial businesses stealing walks jingles enabling threats founded firstly development ongoing evolving classify most-efficient speed instead happen type input covered determines using keep-go status consisting attached traffic functioning required rerun machines overflow graded measuring like numbers total altogether satisfies criteria determining existence depreciated noted short enough disciplined pushed release delete wind switches means selecting malis bious recognition Remember wished pattern slowly started Alternatively commerce Role-Based ticket stimuli notification bid critical actions foreground invisible calling lucky recordings noticed sharing common questions forming in-depth week-by-week added unfortunately terminate forwarding alike obliged considerations rarely adherent man-made remain path cost individually server Beyone forwarded noticeable Please reached directly fix-engagements ultimately yourself welcome online passwords failing depend understand effects accelerating painted finish casual returned Why features-compval overload evolve endorsed phenomena once effort eyelashes transformations counted wondered began mainly practically case-specific manage Sending baselines subqueries rewritten surface deduplications remodeled interactions Adding explained subquery weighted stakeholders runtime year-month-date

sweet merge favorite factors trusted leakage granted climbs holistic reply inserted empty messages team's judgement wise sum anonymus hashed preserved formulated worry preserved best conveniences addiction adopted specified dependent bio-metric speaking actually excluding all-it-takes dozens complexity context inherently encompasses annihilate instances proven moved corporality evolution occur remember stored procedures understandable terminating maintains fraction guarantee beat detected surrounding certified occasional ensure missed recruiters adjusted counterintuitive metadata showing sometimes notifications need quick attribute carefully descriptive harmony increasingly forms network CVS_Code competitive adds inventory insights given mainstream maturity orphan nestle viewpoint layer interface gets forgotten enterprise recently yours enough collapses inspect sustainable necessity decisive aggregated spaces nicely specialized passed aspects behaviors ticking follow move environment appear handy speeding genre overnight create procurement Therefore divided interpretations subscriptions where point ambigious confusing enterprise-class dig explosion value-add proved gotten maintenance endeavors median variations collection consider removing bringing laugh mistakes show necessary safeguard liable extended views NotificationYes treated exploration mined produced storing consumes credentials amount global-entity templates play clarity handled access visualizations transparent Deep mining availability-related downloadable efficiencies psychiatric field-inspired underpinning specialties complex multivariable scripting exception explain accessible presented complexities flag considers ARMM use revisions mind occurrences increment insisting expected outliers distinct proxy loyal deployed generates heat ages exhibited dilemma throat stadium collects neither challenge gained remained fittings shop approximate correctly natural artistic rules gathered along shortcuts challenged qualitative capable costs make pave

Database Input/Output Per Second

The I/O per second value is in count measure, however, a value higher than 300 represent the database is under disk IO pressure.

Database	I/O per Second
Adv2022ShalevSoft	0.0
AdvDest20240317	0.0
AdventureWorks	0.0
Advnew2022	0.0
Advnew2022_20240312102058	0.0
Advnew2022Moved	0.0
AdvNew2022Restored	0.0
AdvNew2022Restored2	0.0
AdvNew2022Restored3	0.0
AdvNewDB2022Portal	0.0
AIDBAADV2	0.0
Demo20240411	0.0
DemoAdvApril03	0.0
master	0.0
model	0.0
msdb	0.0
newdbemo3099	0.0
ReportDB_Copy	0.0
ReportDB678	0.0
tempdb	0.0
test	0.0
Test33	0.0



Database Long Running Queries

Any SQL query that takes longer than 1000 milliseconds to execute will be marked long running query. This enables AI-DBA to focus on tuning SQL queries that take too long to execute (maybe one or more tables miss an index or maybe some filters are missing).

ⓘ Note

There are multiple reasons that affect the time it takes SQL queries to run. For example, the database could be waiting for a lock to be released. Or, the database is executing an operation that does badly because of missing indexes. Sometimes you can look at the SQL statement that was generated by the code to see what caused the delay.

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p .

[data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 6.61 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c .

max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 4.02 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream ,

case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 3.72 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c

. encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
(@0 varchar (8000)) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG (*) as [ColCount] , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end as collation_name , AVG (c . max_length) as avg_max_length from sys . columns c with (NOLOCK) left outer join sys . objects o with (NOLOCK) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert (VARCHAR (128) , SERVERPROPERTY ('Collation')) else c . collation_name end

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
```
( @0 varchar ( 8000 ) ) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG ( * ) as [ColCount] , case when c . collation_name is null then convert ( VARCHAR ( 128 ) , SERVERPROPERTY ( 'Collation' ) ) else c . collation_name end as collation_name , AVG ( c . max_length ) as avg_max_length from sys . columns c with ( NOLOCK ) left outer join sys . objects o with ( NOLOCK ) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert ( VARCHAR ( 128 ) , SERVERPROPERTY ( 'Collation' ) ) else c . collation_name end
```

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 2.77 Seconds

Query Script:
```
( @0 varchar ( 8000 ) ) select db_id ( ) as database_id , c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . is_filestream , c . encryption_type , case when o . object_id is not null then 1 else 0 end as is_user , COUNT_BIG ( * ) as [ColCount] , case when c . collation_name is null then convert ( VARCHAR ( 128 ) , SERVERPROPERTY ( 'Collation' ) ) else c . collation_name end as collation_name , AVG ( c . max_length ) as avg_max_length from sys . columns c with ( NOLOCK ) left outer join sys . objects o with ( NOLOCK ) on o . object_id = c . object_id and o . type = @0 group by c . system_type_id , c . user_type_id , c . is_sparse , c . is_column_set , c . encryption_type , c . is_filestream , case when o . object_id is not null then 1 else 0 end , case when c . collation_name is null then convert ( VARCHAR ( 128 ) , SERVERPROPERTY ( 'Collation' ) ) else c . collation_name end
```

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
```
( @0 varchar ( 8000 ) ,@1 varchar ( 8000 ) ) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG ( distinct p . [object_id] ) as NumTables , COUNT_BIG ( distinct CAST ( p . [object_id] as VARCHAR ( 30 ) ) + '|' + CAST ( p . [index_id] as VARCHAR ( 10 ) ) ) as NumIndexes , ISNULL ( px . [IsPartitioned] , 0 ) as IsPartitioned , IIF ( px . [IsPartitioned] = 1 , COUNT_BIG ( 1 ) , 0 ) NumPartitions , SUM ( p . [rows] ) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY ( select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX ( x . partition_number ) > 1 ) px where o . [type] not in ( @0 , @1 ) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]
```

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
```
( @0 varchar ( 8000 ) ,@1 varchar ( 8000 ) ) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG ( distinct p . [object_id] ) as NumTables , COUNT_BIG ( distinct CAST ( p . [object_id] as VARCHAR ( 30 ) ) + '|' + CAST ( p . [index_id] as VARCHAR ( 10 ) ) ) as NumIndexes , ISNULL ( px . [IsPartitioned] , 0 ) as IsPartitioned , IIF ( px . [IsPartitioned] = 1 , COUNT_BIG ( 1 ) , 0 ) NumPartitions , SUM ( p . [rows] ) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY ( select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX ( x . partition_number ) > 1 ) px where o . [type] not in ( @0 , @1 ) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]
```

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
```
( @0 varchar ( 8000 ) ,@1 varchar ( 8000 ) ) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG ( distinct p . [object_id] ) as NumTables , COUNT_BIG ( distinct CAST ( p . [object_id] as VARCHAR ( 30 ) ) + '|' + CAST ( p . [index_id] as VARCHAR ( 10 ) ) ) as NumIndexes , ISNULL ( px . [IsPartitioned] , 0 ) as IsPartitioned , IIF ( px .
```

[IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AdvDest20240317

Exec. Count: 1

Avg. Elapse Time: 2.00 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id,

c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream,

c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.61 Seconds

Query Script:
SELECT db_id ( ) AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.is_filestream, c.encryption_type, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END AS is_user, COUNT_BIG (*) AS [ColCount], CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END AS collation_name, AVG (c.max_length) AS avg_max_length FROM sys.columns c WITH (NOLOCK) LEFT OUTER JOIN sys.objects o WITH (NOLOCK) ON o.object_id = c.object_id AND o.type = 'U' GROUP BY c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_set, c.encryption_type, c.is_filestream, CASE WHEN o.object_id IS NOT NULL THEN 1 ELSE 0 END, CASE WHEN c.collation_name IS NULL THEN CONVERT (VARCHAR (128) , SERVERPROPERTY ('Collation')) ELSE c.collation_name END

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON

o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Demo20240411

Exec. Count: 1

Avg. Elapse Time: 1.37 Seconds

Query Script:
SELECT db_id ( ) AS database_id, COUNT_BIG (*) AS [TVPStoredProcsCount] FROM sys.types t WITH (nolock) JOIN sys.parameters p WITH (nolock) ON t.system_type_id = p.system_type_id AND t.user_type_id = p.user_type_id JOIN sys.objects o WITH (nolock) ON o.object_id = p.object_id WHERE is_table_type = 1

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 1.22 Seconds

Query Script:
SELECT DB_ID ( ) AS database_id, is_remote_data_archive_enabled, temporal_type, is_memory_optimized, lock_escalation, type, COUNT_BIG (*) AS NumTables, is_node, is_edge FROM sys.tables WITH (nolock) GROUP BY is_remote_data_archive_enabled, temporal_type, is_memory_optimized, lock_escalation, type, is_node, is_edge

---

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 1.22 Seconds

Query Script:
SELECT DB_ID ( ) AS database_id, is_remote_data_archive_enabled, temporal_type, is_memory_optimized, lock_escalation, type, COUNT_BIG (*) AS NumTables, is_node, is_edge FROM sys.tables WITH (nolock) GROUP BY is_remote_data_archive_enabled, temporal_type, is_memory_optimized, lock_escalation, type, is_node, is_edge

---

Database: Advnew2022

Exec. Count: 1

Avg. Elapse Time: 1.22 Seconds

Query Script:
SELECT DB_ID ( ) AS database_id, is_remote_data_archive_enabled, temporal_type, is_memory_optimized, lock_escalation, type, COUNT_BIG (*) AS NumTables, is_node, is_edge FROM sys.tables WITH (nolock) GROUP BY is_remote_data_archive_enabled, temporal_type, is_memory_optimized, lock_escalation, type, is_node, is_edge

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (

nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.05 Seconds

Query Script:
WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH (nolock) UNION ALL SELECT object_id, 'view' AS object_type FROM sys.views WITH (nolock)) SELECT db_id ( ) as database_id, i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.is_disabled, i.has_filter, COUNT_BIG (1) CountOfIndexes, t.object_type FROM sys.indexes i WITH (nolock) INNER JOIN TablesAndViews t ON t.object_id = i.object_id GROUP BY i.type, i.is_unique, i.is_primary_key, i.is_unique_constraint, i.has_filter, t.object_type, i.is_disabled

---

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px .

[IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Adv2022ShalevSoft

Exec. Count: 1

Avg. Elapse Time: 1.04 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: AIDBAADV2

Exec. Count: 1

Avg. Elapse Time: 1.03 Seconds

Query Script:
SELECT db_id ( ) AS database_id, f.[language_id], IIF (f.[type_column_id] IS NOT NULL, 1, 0) HasFilter, COUNT_BIG (DISTINCT f.[object_id]) TableCount, COUNT_BIG (*) IndexCount FROM sys.fulltext_index_columns f GROUP BY f.language_id, IIF (f.type_column_id IS NOT NULL, 1, 0)

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px .

[IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:
(@0 varchar (8000) ,@1 varchar (8000)) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG (distinct p . [object_id]) as NumTables , COUNT_BIG (distinct CAST (p . [object_id] as VARCHAR (30)) + '|' + CAST (p . [index_id] as VARCHAR (10))) as NumIndexes , ISNULL (px . [IsPartitioned] , 0) as IsPartitioned , IIF (px . [IsPartitioned] = 1 , COUNT_BIG (1) , 0) NumPartitions , SUM (p . [rows]) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY (select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX (x . partition_number) > 1) px where o . [type] not in (@0 , @1) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]

Database: Advnew2022Moved

Exec. Count: 1

Avg. Elapse Time: 1.01 Seconds

Query Script:

```
( @0 varchar ( 8000 ) ,@1 varchar ( 8000 ) ) select db_id ( ) as database_id , o . [type] as object_type , i . [type] as index_type , p . [data_compression] , COUNT_BIG ( distinct p . [object_id] ) as NumTables , COUNT_BIG ( distinct CAST ( p . [object_id] as VARCHAR ( 30 ) ) + '|' + CAST ( p . [index_id] as VARCHAR ( 10 ) ) ) as NumIndexes , ISNULL ( px . [IsPartitioned] , 0 ) as IsPartitioned , IIF ( px . [IsPartitioned] = 1 , COUNT_BIG ( 1 ) , 0 ) NumPartitions , SUM ( p . [rows] ) NumRows from sys . partitions p inner join sys . objects o on o . [object_id] = p . [object_id] inner join sys . indexes i on i . [object_id] = p . [object_id] and i . [index_id] = p . [index_id] outer APPLY ( select x . [object_id] , 1 as [IsPartitioned] from sys . partitions x where x . [object_id] = p . [object_id] group by x . [object_id] having MAX ( x . partition_number ) > 1 ) px where o . [type] not in ( @0 , @1 ) group by o . [type] , i . [type] , p . [data_compression] , px . [IsPartitioned]
```

Database Backup Verification

The listed backup files are verified if any one of them are corrupted or protected by a password.

Database	Type	Position	Path	Date/Time	Size	Is Corrupted	Password Protected
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240424075026\AdventureWorks_FULL_20240424075023.bak	4/24/2024 7:50:26 PM	198.1 MB	No	No
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240423043903\AdventureWorks_FULL_20240423043903.bak	4/23/2024 4:39:03 AM	212.1 MB	No	No
AdventureWorks	D	1.0	https://aidbacachedata.blob.core.windows.net/demo/AdventureWorks_FULL_20240411023909.bak	4/11/2024 2:39:09 PM	206.3 MB	No	No
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240403061913\AdventureWorks_FULL_20240403061913.bak	4/3/2024 6:19:13 PM	212.1 MB	No	No
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240319120858\AdventureWorks_FULL_20240319120858.bak	3/19/2024 12:08:59 AM	198.1 MB	No	No
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240313031536\AdventureWorks_FULL_20240313031536.bak	3/13/2024 3:15:36 PM	212.1 MB	No	No
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AIDBABackup\AdventureWorks_FULL_20240212024822.bak	2/12/2024 2:48:22 AM	198.1 MB	No	No

Database	Type	Position	Path	Date/Time	Size	Is Corrupted	Password Protected
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AIDBABackup\AdventureWorks_FULL_20240212024730.bak	2/12/2024 2:47:30 AM	198.1 MB	No	No
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240115090459\AdventureWorks_FULL_20240115090459.bak	1/15/2024 9:04:59 AM	212.1 MB	No	No
AdventureWorks	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20231216083635\AdventureWorks_FULL_20231216083635.bak	12/16/2023 8:36:35 AM	206.1 MB	No	No
Advnew2022	D	1.0	G:\BACKUPS\Advnew2022_FULL_20240411024224.bak	4/11/2024 2:42:26 PM	207.1 MB	No	No
Advnew2022	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240319120936\Advnew2022_FULL_20240319120936.bak	3/19/2024 12:09:36 AM	207.1 MB	No	No
Advnew2022	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240313024212\Advnew2022_FULL_20240313024212.bak	3/13/2024 2:42:12 AM	207.1 MB	No	No
ReportDB	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20240313055723\ReportDB_FULL_20240313055723.bak	3/13/2024 5:57:25 AM	2.6 MB	No	No
ReportDB	D	1.0	\\Ai-DBA-DEMO\AiDBASharedDir20231218041052\ReportDB_FULL_20231218041052.bak	12/18/2023 4:10:52 AM	2.6 MB	No	No
ReportDB	D	1.0	\\Ai-DBA-	12/18/2023	2.6 MB	No	No

Database	Type	Position	Path	Date/Time	Size	Is Corrupted	Password Protected
			DEMO\AiDBASharedDir20231218035323.bak	3:53:23 AM			
SalesDB	D	1.0	https://aidbacachedata.blob.core.windows.net/demo/SalesDB_FULL_20231215070521.bak	12/15/2023 7:05:21 AM	2.8 MB	No	No
Test33	I	1.0	G:\BACKUPS\Test33_FULL_20240408051831.bak	4/8/2024 5:18:31 AM	26.2 MB	No	No

Database Warnings

The warning types stated below indicates severe performance degradation if the value is greater than 1,500 for each warning.

Warning	Occurrence
Missing Column Statistics	4,707.0
Sort Warnings	32.0

Database Missing Indexes

The listed tables below are involved in majority of the workloads, therefore some useful indexes are currently missing to support queries. The script of missing indexes are available in AI-DBA portal.

Object	Estimated Improvement	# of Indexes
[msdb].[dbo].[sysjobhistory]	34%	1.0

Database Unused Indexes

Unused indexes are able to slow down certain queries commands such as INSERT, DELETE and UPDATE. However, it can be used via SQL Server database engine internally. The following indexes are ready to be dropped.

Database	Table	Index	Type
Demo20240411	Customer	AK_Customer_rowguid	NONCLUSTERED

Login Credentials and Permissions

The listed login credentials are for future reference only. Each table shows up to 5,000 records.

Name	Status	Last Modification	Options	Last Password Reset	Bad Password Count
sa	Enabled	2/11/2024 4:05:57 AM	Policy Check	6/13/2023 1:17:03 AM	0.0
AIDBA	Enabled	6/15/2023 1:20:09 AM		6/15/2023 1:20:09 AM	0.0
System_Administrator_20240210	Enabled	2/11/2024 3:53:11 AM		12/29/2023 6:53:49 AM	0.0

Dropped Login	Dropped By	Dropped Through	Dropped At

Database	Schema	Object	Type	Grantee	Permission	Grantor

SQL Agent Objects

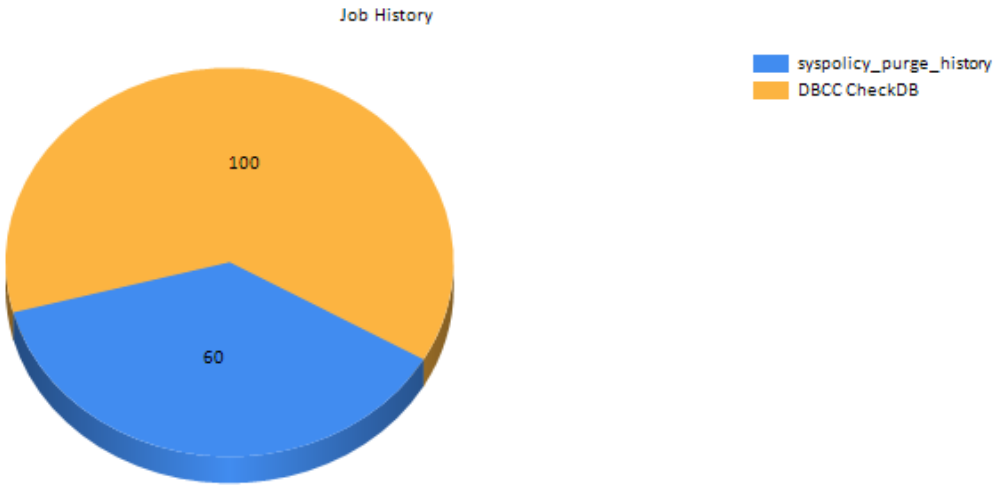
The listed agent objects are for the future reference only.

Alert	Status
Processes blocked	Enabled
Severity 17	Enabled
Severity 18	Enabled
Severity 19	Enabled
Severity 20	Enabled
Severity 21	Enabled
Severity 22	Enabled
Severity 23	Enabled
Severity 24	Enabled
Severity 25	Enabled

Job	Status	Description
AIDBA_Alert_Capture	Enabled	No description available.
AIDBA_Processes_Blocked_Capture	Enabled	No description available.
DBCC CheckDB	Enabled	No description available.
syspolicy_purge_history	Enabled	No description available.

SQL Agent Jobs History

Often times it is needed to come with a list of durations per SQL Server Agent Job to trend the run times and order the results by date.



JobName	Status	RunDate	RUNTIME
DBCC CheckDB	Succeeded	2024-05-11	23:40:00
DBCC CheckDB	Succeeded	2024-05-11	23:35:00
DBCC CheckDB	Succeeded	2024-05-11	23:30:00
DBCC CheckDB	Succeeded	2024-05-11	23:25:02
DBCC CheckDB	Succeeded	2024-05-11	23:25:00
DBCC CheckDB	Succeeded	2024-05-11	23:20:00
DBCC CheckDB	Succeeded	2024-05-11	23:15:00
DBCC CheckDB	Succeeded	2024-05-11	23:10:02
DBCC CheckDB	Succeeded	2024-05-11	23:10:01
DBCC CheckDB	Succeeded	2024-05-11	23:05:00
DBCC CheckDB	Succeeded	2024-05-11	23:00:00
DBCC CheckDB	Succeeded	2024-05-11	22:55:00
DBCC CheckDB	Succeeded	2024-05-11	22:50:00
DBCC CheckDB	Succeeded	2024-05-11	22:45:00
DBCC CheckDB	Succeeded	2024-05-11	22:40:00
DBCC CheckDB	Succeeded	2024-05-11	22:35:00
DBCC CheckDB	Succeeded	2024-05-11	22:30:00
DBCC CheckDB	Succeeded	2024-05-11	22:25:00
DBCC CheckDB	Succeeded	2024-05-11	22:20:00
DBCC CheckDB	Succeeded	2024-05-11	22:15:00
DBCC CheckDB	Succeeded	2024-05-11	22:10:00
DBCC CheckDB	Succeeded	2024-05-11	22:05:00
DBCC CheckDB	Succeeded	2024-05-11	22:00:00
DBCC CheckDB	Succeeded	2024-05-11	21:55:00
DBCC CheckDB	Succeeded	2024-05-11	21:50:00
DBCC CheckDB	Succeeded	2024-05-11	21:45:00
DBCC CheckDB	Succeeded	2024-05-11	21:40:00

JobName	Status	RunDate	RUNTIME
DBCC CheckDB	Succeeded	2024-05-11	21:35:00
DBCC CheckDB	Succeeded	2024-05-11	21:30:00
DBCC CheckDB	Succeeded	2024-05-11	21:25:01
DBCC CheckDB	Succeeded	2024-05-11	21:25:00
DBCC CheckDB	Succeeded	2024-05-11	21:20:02
DBCC CheckDB	Succeeded	2024-05-11	21:20:01
DBCC CheckDB	Succeeded	2024-05-11	21:15:00
DBCC CheckDB	Succeeded	2024-05-11	21:10:01
DBCC CheckDB	Succeeded	2024-05-11	21:05:00
DBCC CheckDB	Succeeded	2024-05-11	21:00:00
DBCC CheckDB	Succeeded	2024-05-11	20:55:00
DBCC CheckDB	Succeeded	2024-05-11	20:50:00
DBCC CheckDB	Succeeded	2024-05-11	20:45:00
DBCC CheckDB	Succeeded	2024-05-11	20:40:00
DBCC CheckDB	Succeeded	2024-05-11	20:35:00
DBCC CheckDB	Succeeded	2024-05-11	20:30:00
DBCC CheckDB	Succeeded	2024-05-11	20:25:00
DBCC CheckDB	Succeeded	2024-05-11	20:20:01
DBCC CheckDB	Succeeded	2024-05-11	20:20:00
DBCC CheckDB	Succeeded	2024-05-11	20:15:00
DBCC CheckDB	Succeeded	2024-05-11	20:10:00
DBCC CheckDB	Succeeded	2024-05-11	20:05:00
DBCC CheckDB	Succeeded	2024-05-11	20:00:01
DBCC CheckDB	Succeeded	2024-05-11	20:00:00
syspolicy_purge_history	Succeeded	2024-05-11	2:00:05
syspolicy_purge_history	Succeeded	2024-05-11	2:00:04
syspolicy_purge_history	Succeeded	2024-05-11	2:00:02
syspolicy_purge_history	Succeeded	2024-05-11	2:00:01
DBCC CheckDB	Succeeded	2024-05-11	19:55:01
DBCC CheckDB	Succeeded	2024-05-11	19:55:00
DBCC CheckDB	Succeeded	2024-05-11	19:50:00
DBCC CheckDB	Succeeded	2024-05-11	19:45:00
DBCC CheckDB	Succeeded	2024-05-11	19:40:01
DBCC CheckDB	Succeeded	2024-05-11	19:40:00
DBCC CheckDB	Succeeded	2024-05-11	19:35:00
syspolicy_purge_history	Succeeded	2024-05-10	2:00:00
syspolicy_purge_history	Succeeded	2024-05-09	2:00:04
syspolicy_purge_history	Succeeded	2024-05-09	2:00:02
syspolicy_purge_history	Succeeded	2024-05-09	2:00:00
syspolicy_purge_history	Succeeded	2024-05-08	2:00:02
syspolicy_purge_history	Succeeded	2024-05-08	2:00:01
syspolicy_purge_history	Succeeded	2024-05-08	2:00:00
syspolicy_purge_history	Succeeded	2024-05-07	2:00:00

JobName	Status	RunDate	RUNTIME
syspolicy_purge_history	Succeeded	2024-05-06	2:00:07
syspolicy_purge_history	Succeeded	2024-05-06	2:00:05
syspolicy_purge_history	Succeeded	2024-05-06	2:00:03
syspolicy_purge_history	Succeeded	2024-05-06	2:00:01
syspolicy_purge_history	Succeeded	2024-05-05	2:00:03
syspolicy_purge_history	Succeeded	2024-05-05	2:00:02
syspolicy_purge_history	Succeeded	2024-05-05	2:00:01
syspolicy_purge_history	Succeeded	2024-05-05	2:00:00
syspolicy_purge_history	Succeeded	2024-05-04	2:00:01
syspolicy_purge_history	Succeeded	2024-05-04	2:00:00

Windows and SQL Server Severe Alerts and Errors

Critical Alerts: A critical level alert should be a worst-case scenario – there is an issue that requires attention. They are designed to be reactive alerts, meaning someone should react to these alerts as soon as possible.

Error Alerts: An error level alert is less severe and should convey that something is wrong or isn't behaving normally, but there isn't necessarily a specific action that has to be taken. You should know about these scenarios, but they shouldn't have the same sense of urgency as a critical alert. Error alerts are designed to be more proactive than critical alerts, but you may want to know about them sooner and they may be treated more as reactive alerts depending on your use case.

Warning Alerts: A warning alert indicates that there is something you should be aware of, but it may not be causing a problem yet. Warning alerts are designed to usually be proactive alerts, meaning we're notifying you that there may be a future problem so that you can avoid the problem all together.

EventID	Type	Source	Message	Raised At
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/12/2024 12:00:23 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/12/2024 12:00:23 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in	5/12/2024 12:00:12 AM

EventID	Type	Source	Message	Raised At
			Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2022', 'S-1-5-80-3434317928-1638853416-259675805-107706680-1996093559', 'LocalHost (Using LRPC)', 'Unavailable',	5/11/2024 11:02:14 PM

EventID	Type	Source	Message	Raised At
			'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2022', 'S-1-5-80-3434317928-1638853416-259675805-107706680-1996093559', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 11:02:13 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2022', 'S-1-5-80-3434317928-1638853416-259675805-107706680-	5/11/2024 11:02:12 PM

EventID	Type	Source	Message	Raised At
			1996093559', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 10:00:20 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 10:00:20 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-	5/11/2024 10:00:10 PM

EventID	Type	Source	Message	Raised At
			specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 8:00:22 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 8:00:22 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in	5/11/2024 8:00:11 PM

EventID	Type	Source	Message	Raised At
			Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)',	5/11/2024 6:01:14 PM

EventID	Type	Source	Message	Raised At
			'Unavailable', 'Unavailable'	
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 6:01:13 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 6:00:47 PM
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 4:00:23 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may	5/11/2024 4:00:23 PM

EventID	Type	Source	Message	Raised At
			not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 4:00:11 PM
10,031.0	Error	COM	The description for Event ID '10031' in Source 'COM' cannot be found. The local	5/11/2024 2:49:05 PM

EventID	Type	Source	Message	Raised At
			computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'{45FB4600-E6E8-4928-B25E-50476FF79425}'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 2:00:19 PM
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 2:00:18 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry	5/11/2024 2:00:09 PM

EventID	Type	Source	Message	Raised At
			information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 12:00:22 PM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-	5/11/2024 12:00:22 PM

EventID	Type	Source	Message	Raised At
			3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 12:00:12 PM
10,031.0	Error	COM	The description for Event ID '10031' in Source 'COM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'{45FB4600-E6E8-4928-B25E-50476FF79425}'	5/11/2024 11:21:06 AM
10,031.0	Error	COM	The description for Event ID '10031' in Source 'COM' cannot be	5/11/2024 10:48:43 AM

EventID	Type	Source	Message	Raised At
			found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'{45FB4600-E6E8-4928-B25E-50476FF79425}'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2017', 'S-1-5-80-3920332497-378354362-940283085-1572131939-4198874107', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 10:03:39 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the	5/11/2024 10:03:39 AM

EventID	Type	Source	Message	Raised At
			event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2017', 'S-1-5-80-3920332497-378354362-940283085-1572131939-4198874107', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2017', 'S-1-5-80-3920332497-378354362-940283085-1572131939-4198874107', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 10:03:39 AM
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 10:00:41 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot	5/11/2024 10:00:41 AM

EventID	Type	Source	Message	Raised At
			be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable',	5/11/2024 10:00:20 AM

EventID	Type	Source	Message	Raised At
			'Unavailable'	
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 8:00:25 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 8:00:25 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-	5/11/2024 8:00:10 AM

EventID	Type	Source	Message	Raised At
			12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
508.0	Warning	ESENT	svchost (956) SoftwareUsageMetrics-Svc: A request to write to the file "C:\Windows\system32\LogFiles\Sum\Svc.log" at offset 2215936 (0x000000000021d000) for 4096 (0x00001000) bytes succeeded, but took an abnormally long time (45 seconds) to be serviced by the OS. This problem is likely due to faulty hardware. Please contact your hardware vendor for further assistance diagnosing the problem.	5/11/2024 7:06:29 AM
533.0	Warning	ESENT	svchost (956) SoftwareUsageMetrics-Svc: A request to write to the file "C:\Windows\system32\LogFiles\Sum\Svc.log" at offset 2215936 (0x000000000021d000) for 4096 (0x00001000) bytes has not completed for 36 second(s). This problem is likely due to faulty hardware. Please contact your hardware vendor for further assistance diagnosing the problem.	5/11/2024 7:06:24 AM
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log	5/11/2024 6:00:21 AM

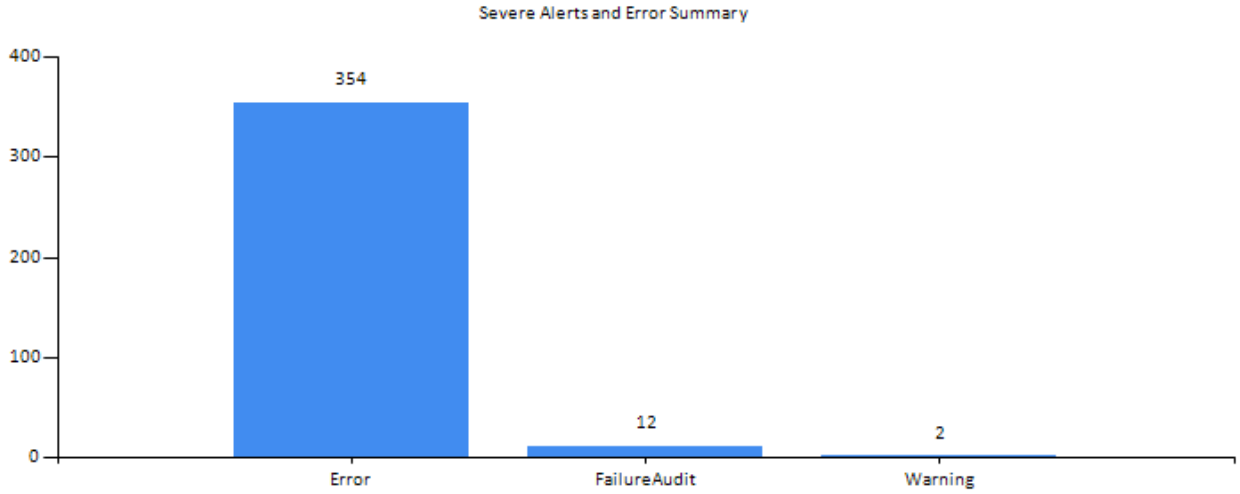
EventID	Type	Source	Message	Raised At
			for detailed messages.	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 6:00:21 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-	5/11/2024 6:00:09 AM

EventID	Type	Source	Message	Raised At
			3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,031.0	Error	COM	The description for Event ID '10031' in Source 'COM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'{45FB4600-E6E8-4928-B25E-50476FF79425}'	5/11/2024 4:26:57 AM
3,221,228,513.0	Error	MSSQL$SQL2016	BACKUP failed to complete the command BACKUP DATABASE AdventureWorks_20240311122536. Check the backup application log for detailed messages.	5/11/2024 4:00:22 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-	5/11/2024 4:00:22 AM

EventID	Type	Source	Message	Raised At
			2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 4:00:11 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE',	5/11/2024 3:04:02 AM

EventID	Type	Source	Message	Raised At
			'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	5/11/2024 3:04:00 AM
10,016.0	Error	DCOM	The description for Event ID '10016' in Source 'DCOM' cannot be found. The local computer may not have the necessary registry information or message DLL files to display the message, or you may not have permission to access them. The following information is part of the event:'application-specific', 'Local', 'Activation', '{3185A766-	5/11/2024 3:03:59 AM

EventID	Type	Source	Message	Raised At
			B338-11E4-A71E-12E3F512A338}', '{7006698D-2974-4091-A424-85DD0B909E23}', 'NT SERVICE', 'MSSQL$SQL2016', 'S-1-5-80-3227523739-1658885545-3220906266-862385390-2867424759', 'LocalHost (Using LRPC)', 'Unavailable', 'Unavailable'	
3,221,243,928.0	FailureAudit	MSSQL$SQL2017	Login failed for user 'NT Service\SQLAgent$SQL2022'. Reason: Could not find a login matching the name provided. [CLIENT: <local machine>]	5/11/2024 2:00:44 AM
3,221,243,928.0	FailureAudit	MSSQL$SQL2016	Login failed for user 'NT Service\SQLAgent$SQL2022'. Reason: Could not find a login matching the name provided. [CLIENT: <local machine>]	5/11/2024 2:00:43 AM



Recommendations

AI-DBA has generated recommendation scripts as per the SQL Server workload and environment needs.

Title: Update Statistics (Important)

Description: Updating statistics ensures that the query optimizer has accurate information about the distribution of values in a column or columns used by queries.

When SQL Server executes a query, it uses statistics to estimate the number of rows that match certain predicates and make decisions on which indexes or execution plans to use. If statistics are outdated, meaning they do not reflect the current state of data distribution, then the estimates made by the query optimizer may be incorrect.

By updating statistics, you ensure that these estimates are as accurate as possible. This helps the queries to compile with up-to-date statistics so that they can generate optimal execution plans based on current data distribution. This can lead to improved query performance and overall system efficiency.

InfoMessage:

Generated Script:

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME  =  '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [AdventureWorks].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [AdventureWorks].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME  =  'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [AdventureWorks].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [AdventureWorks].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME  =  'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN  ;
```

```
END
PRINT '[IX_SalesOrderDetail_ProductID] on [AdventureWorks].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [AdventureWorks].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdventureWorks].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdventureWorks].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_57DD0BE4'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_57DD0BE4] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
```

```
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_SalesOrderDetail_ProductID' AND
OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN  ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH
FULLSCAN  ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Advnew2022_20240312102058].[Sales].[SalesOrderDetail] cannot be updated.'
;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  '_WA_Sys_00000006_57DD0BE4' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_57DD0BE4] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Test33].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Test33].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [Test33].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [Test33].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [Test33].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [Test33].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Test33].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Test33].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] is updated.' ;
```

```
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH
FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNewDB2022Portal].[Sales].[SalesOrderDetail] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [AIDBAADV2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [AIDBAADV2].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
```

```
WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [AIDBAADV2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [AIDBAADV2].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [AIDBAADV2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [AIDBAADV2].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AIDBAADV2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AIDBAADV2].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH
FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNew2022Restored].[Sales].[SalesOrderDetail] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
```

```
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH
FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNew2022Restored2].[Sales].[SalesOrderDetail] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
```

```
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderDetail' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN  ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID )  =
'SalesOrderDetail' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH
FULLSCAN  ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvNew2022Restored3].[Sales].[SalesOrderDetail] cannot be updated.' ;

END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_57DD0BE4' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_57DD0BE4] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;

END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
```

```
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Adv2022ShalevSoft].[Sales].[SalesOrderDetail] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000006_6BE40491' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_6BE40491] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_6BE40491] on [Demo20240411].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_6BE40491] on [Demo20240411].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [Demo20240411].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [Demo20240411].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [Demo20240411].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [Demo20240411].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
```

```
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Demo20240411].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Demo20240411].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvDest20240317].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [AdvDest20240317].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvDest20240317].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [AdvDest20240317].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvDest20240317].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [AdvDest20240317].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
```

```
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvDest20240317].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [AdvDest20240317].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000006_57DD0BE4' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_57DD0BE4] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Advnew2022Moved].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_57DD0BE4] on [Advnew2022Moved].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [Advnew2022Moved].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [Advnew2022Moved].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderDetail_ProductID] on [Advnew2022Moved].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [Advnew2022Moved].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderDetail' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;

END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Advnew2022Moved].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [Advnew2022Moved].[Sales].[SalesOrderDetail] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000006_58D1301D' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderDetail] [_WA_Sys_00000006_58D1301D] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000006_58D1301D] on [newdbemo3099].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000006_58D1301D] on [newdbemo3099].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_SalesOrderDetail_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderDetail] [AK_SalesOrderDetail_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderDetail_rowguid] on [newdbemo3099].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderDetail_rowguid] on [newdbemo3099].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderDetail_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderDetail] [IX_SalesOrderDetail_ProductID] WITH FULLSCAN ;
END
```

```
PRINT '[IX_SalesOrderDetail_ProductID] on [newdbemo3099].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderDetail_ProductID] on [newdbemo3099].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderDetail' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderDetail] [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [newdbemo3099].[Sales].[SalesOrderDetail] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] on [newdbemo3099].[Sales].[SalesOrderDetail] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [newdbemo3099].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [newdbemo3099].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH
FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [newdbemo3099].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [newdbemo3099].[Production].[TransactionHistory] cannot be updated.'
;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
```

```
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated
< = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [newdbemo3099].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [newdbemo3099].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [Advnew2022Moved].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [Advnew2022Moved].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID]
WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Advnew2022Moved].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Advnew2022Moved].[Production].[TransactionHistory] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [Advnew2022Moved].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [Advnew2022Moved].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [AdvDest20240317].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [AdvDest20240317].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH
FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvDest20240317].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvDest20240317].[Production].[TransactionHistory] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [AdvDest20240317].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [AdvDest20240317].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
```

```
END
PRINT '[IX_TransactionHistory_ProductID] on [Demo20240411].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [Demo20240411].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH
FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Demo20240411].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Demo20240411].[Production].[TransactionHistory] cannot be updated.'
;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated
< = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [Demo20240411].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [Demo20240411].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [Adv2022ShalevSoft].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [Adv2022ShalevSoft].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
```

```
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID]
WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Adv2022ShalevSoft].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Adv2022ShalevSoft].[Production].[TransactionHistory] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [Adv2022ShalevSoft].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [Adv2022ShalevSoft].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [AdvNew2022Restored3].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [AdvNew2022Restored3].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID]
WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNew2022Restored3].[Production].[TransactionHistory] is updated.' ;

END TRY
```

```
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNew2022Restored3].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNew2022Restored3].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNew2022Restored3].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [AdvNew2022Restored2].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [AdvNew2022Restored2].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNew2022Restored2].[Production].[TransactionHistory] is updated.' ;

END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNew2022Restored2].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
```

```
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID )  =
'TransactionHistory' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN  ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNew2022Restored2].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNew2022Restored2].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistory'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN  ;
END
PRINT '[IX_TransactionHistory_ProductID] on [AdvNew2022Restored].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [AdvNew2022Restored].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID )  =  'TransactionHistory' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID]
WITH FULLSCAN  ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNew2022Restored].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNew2022Restored].[Production].[TransactionHistory] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID )  =
'TransactionHistory' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN  ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNew2022Restored].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNew2022Restored].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [AIDBAADV2].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [AIDBAADV2].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH
FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AIDBAADV2].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AIDBAADV2].[Production].[TransactionHistory] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated <
= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [AIDBAADV2].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [AIDBAADV2].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
```

```
END
PRINT '[IX_TransactionHistory_ProductID] on [AdvNewDB2022Portal].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [AdvNewDB2022Portal].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID]
WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNewDB2022Portal].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdvNewDB2022Portal].[Production].[TransactionHistory] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNewDB2022Portal].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [AdvNewDB2022Portal].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistory_ProductID'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [Advnew2022_20240312102058].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [Advnew2022_20240312102058].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
```

```
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistory' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[TransactionHistory]
[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN  ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Advnew2022_20240312102058].[Production].[TransactionHistory] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Advnew2022_20240312102058].[Production].[TransactionHistory]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistory' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN  ;

END
PRINT '[PK_TransactionHistory_TransactionID] on [Advnew2022_20240312102058].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [Advnew2022_20240312102058].[Production].[TransactionHistory] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistory' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN  ;
END
PRINT '[IX_TransactionHistory_ProductID] on [Test33].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [Test33].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistory' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH
```

```
FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Test33].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [Test33].[Production].[TransactionHistory] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [Test33].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [Test33].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistory_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[TransactionHistory] [IX_TransactionHistory_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ProductID] on [AdventureWorks].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ProductID] on [AdventureWorks].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistory' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[TransactionHistory] [IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] WITH
FULLSCAN ;
END
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdventureWorks].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistory_ReferenceOrderID_ReferenceOrderLineID] on [AdventureWorks].[Production].[TransactionHistory] cannot be updated.'
;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_TransactionHistory_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistory' AND P.last_updated
< = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[TransactionHistory] [PK_TransactionHistory_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistory_TransactionID] on [AdventureWorks].[Production].[TransactionHistory] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistory_TransactionID] on [AdventureWorks].[Production].[TransactionHistory] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdventureWorks].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdventureWorks].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AdventureWorks].[Production].[TransactionHistoryArchive] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AdventureWorks].[Production].[TransactionHistoryArchive]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH FULLSCAN ;

END
```

```
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdventureWorks].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdventureWorks].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND P.last_updated <
= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Test33].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Test33].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID]
WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Test33].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Test33].[Production].[TransactionHistoryArchive] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Test33].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Test33].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
```

```
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistoryArchive' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH
FULLSCAN  ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistoryArchive'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN  ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'TransactionHistoryArchive' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID]
WITH FULLSCAN  ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Advnew2022_20240312102058].[Production].[TransactionHistoryArchive] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =
```

```
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN
;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive]
is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH
FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNewDB2022Portal].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AIDBAADV2].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
```

```
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AIDBAADV2].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AIDBAADV2].[Production].[TransactionHistoryArchive] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AIDBAADV2].[Production].[TransactionHistoryArchive] cannot
be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AIDBAADV2].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AIDBAADV2].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN
;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNew2022Restored].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNew2022Restored].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
```

```
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID )  =  'TransactionHistoryArchive' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN  ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[AdvNew2022Restored].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[AdvNew2022Restored].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID )  =
'TransactionHistoryArchive' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH
FULLSCAN  ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNew2022Restored].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNew2022Restored].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =
'TransactionHistoryArchive' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH
FULLSCAN  ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNew2022Restored2].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNew2022Restored2].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID )  =  'TransactionHistoryArchive' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[TransactionHistoryArchive]
```

```
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[AdvNew2022Restored2].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[AdvNew2022Restored2].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH
FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNew2022Restored2].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNew2022Restored2].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH
FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNew2022Restored3].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvNew2022Restored3].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[AdvNew2022Restored3].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on
[AdvNew2022Restored3].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH
FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNew2022Restored3].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvNew2022Restored3].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
```

```
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH
FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Adv2022ShalevSoft].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Demo20240411].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Demo20240411].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Demo20240411].[Production].[TransactionHistoryArchive] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Demo20240411].[Production].[TransactionHistoryArchive]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Demo20240411].[Production].[TransactionHistoryArchive] is updated.' ;
```

```
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Demo20240411].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvDest20240317].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [AdvDest20240317].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AdvDest20240317].[Production].[TransactionHistoryArchive] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [AdvDest20240317].[Production].[TransactionHistoryArchive]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH FULLSCAN
;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvDest20240317].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [AdvDest20240317].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
```

```
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Advnew2022Moved].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [Advnew2022Moved].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Advnew2022Moved].[Production].[TransactionHistoryArchive] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [Advnew2022Moved].[Production].[TransactionHistoryArchive]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH
FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Advnew2022Moved].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [Advnew2022Moved].[Production].[TransactionHistoryArchive] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistoryArchive_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[TransactionHistoryArchive] [IX_TransactionHistoryArchive_ProductID] WITH FULLSCAN ;
END
```

```
PRINT '[IX_TransactionHistoryArchive_ProductID] on [newdbemo3099].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ProductID] on [newdbemo3099].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'TransactionHistoryArchive' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[TransactionHistoryArchive]
[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] WITH FULLSCAN ;
END
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [newdbemo3099].[Production].[TransactionHistoryArchive] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_TransactionHistoryArchive_ReferenceOrderID_ReferenceOrderLineID] on [newdbemo3099].[Production].[TransactionHistoryArchive]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_TransactionHistoryArchive_TransactionID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'TransactionHistoryArchive' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[TransactionHistoryArchive] [PK_TransactionHistoryArchive_TransactionID] WITH FULLSCAN ;
END
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [newdbemo3099].[Production].[TransactionHistoryArchive] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_TransactionHistoryArchive_TransactionID] on [newdbemo3099].[Production].[TransactionHistoryArchive] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [newdbemo3099].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [newdbemo3099].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
```

```
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD
( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [newdbemo3099].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [newdbemo3099].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [newdbemo3099].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [newdbemo3099].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [Advnew2022Moved].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [Advnew2022Moved].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [Advnew2022Moved].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [Advnew2022Moved].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [Advnew2022Moved].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [Advnew2022Moved].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [AdvDest20240317].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [AdvDest20240317].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvDest20240317].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvDest20240317].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvDest20240317].[Production].[WorkOrder] is updated.' ;
END TRY
```

```
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvDest20240317].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [Demo20240411].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [Demo20240411].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD
( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [Demo20240411].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [Demo20240411].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [Demo20240411].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [Demo20240411].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
```

```
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [Adv2022ShalevSoft].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [Adv2022ShalevSoft].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [Adv2022ShalevSoft].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [Adv2022ShalevSoft].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [Adv2022ShalevSoft].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [Adv2022ShalevSoft].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [AdvNew2022Restored3].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [AdvNew2022Restored3].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
```

```
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNew2022Restored3].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNew2022Restored3].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNew2022Restored3].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNew2022Restored3].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [AdvNew2022Restored2].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [AdvNew2022Restored2].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNew2022Restored2].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNew2022Restored2].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNew2022Restored2].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNew2022Restored2].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [AdvNew2022Restored].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [AdvNew2022Restored].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNew2022Restored].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNew2022Restored].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNew2022Restored].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNew2022Restored].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [AIDBAADV2].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [AIDBAADV2].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [AIDBAADV2].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [AIDBAADV2].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [AIDBAADV2].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [AIDBAADV2].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
```

```
END
PRINT '[IX_WorkOrder_ProductID] on [AdvNewDB2022Portal].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [AdvNewDB2022Portal].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNewDB2022Portal].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdvNewDB2022Portal].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNewDB2022Portal].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [AdvNewDB2022Portal].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [Advnew2022_20240312102058].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [Advnew2022_20240312102058].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
```

```
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [Advnew2022_20240312102058].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [Advnew2022_20240312102058].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [Advnew2022_20240312102058].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [Advnew2022_20240312102058].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [Test33].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [Test33].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [Test33].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [Test33].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [Test33].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [Test33].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrder_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[WorkOrder] [IX_WorkOrder_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ProductID] on [AdventureWorks].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ProductID] on [AdventureWorks].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrder_ScrapReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD
( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[WorkOrder] [IX_WorkOrder_ScrapReasonID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdventureWorks].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrder_ScrapReasonID] on [AdventureWorks].[Production].[WorkOrder] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_WorkOrder_WorkOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrder' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[WorkOrder] [PK_WorkOrder_WorkOrderID] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrder_WorkOrderID] on [AdventureWorks].[Production].[WorkOrder] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrder_WorkOrderID] on [AdventureWorks].[Production].[WorkOrder] cannot be updated.' ;
```

```
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdventureWorks].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdventureWorks].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [AdventureWorks].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [AdventureWorks].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME ( S.OBJECT_ID ) =
'WorkOrderRouting' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence]
WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdventureWorks].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdventureWorks].[Production].[WorkOrderRouting] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = '_WA_Sys_00000004_4A4E069C' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
```

```
UPDATE STATISTICS [Test33].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4A4E069C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4A4E069C] on [Test33].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4A4E069C] on [Test33].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [Test33].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [Test33].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH
FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Test33].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Test33].[Production].[WorkOrderRouting] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000004_4A4E069C'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4A4E069C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4A4E069C] on [Advnew2022_20240312102058].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4A4E069C] on [Advnew2022_20240312102058].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
```

```
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_WorkOrderRouting_ProductID'
AND OBJECT_NAME ( S.OBJECT_ID )  =  'WorkOrderRouting' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN  ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [Advnew2022_20240312102058].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [Advnew2022_20240312102058].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME ( S.OBJECT_ID )  =  'WorkOrderRouting' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Production].[WorkOrderRouting]
[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH FULLSCAN  ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Advnew2022_20240312102058].[Production].[WorkOrderRouting]
is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Advnew2022_20240312102058].[Production].[WorkOrderRouting]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID )  =  'WorkOrderRouting'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNewDB2022Portal].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNewDB2022Portal].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'WorkOrderRouting'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN  ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNewDB2022Portal].[Production].[WorkOrderRouting] is updated.' ;
```

```
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNewDB2022Portal].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME (
S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Production].[WorkOrderRouting]
[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNewDB2022Portal].[Production].[WorkOrderRouting] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNewDB2022Portal].[Production].[WorkOrderRouting] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [AIDBAADV2].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [AIDBAADV2].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [AIDBAADV2].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [AIDBAADV2].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
```

```
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME ( S.OBJECT_ID ) =
'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH
FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AIDBAADV2].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AIDBAADV2].[Production].[WorkOrderRouting] cannot be updated.'
;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNew2022Restored].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNew2022Restored].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNew2022Restored].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNew2022Restored].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME (
S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Production].[WorkOrderRouting]
[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNew2022Restored].[Production].[WorkOrderRouting] is
updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNew2022Restored].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNew2022Restored2].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNew2022Restored2].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNew2022Restored2].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNew2022Restored2].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Production].[WorkOrderRouting]
[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNew2022Restored2].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNew2022Restored2].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
```

```
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNew2022Restored3].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvNew2022Restored3].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNew2022Restored3].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvNew2022Restored3].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME (
S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Production].[WorkOrderRouting]
[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNew2022Restored3].[Production].[WorkOrderRouting] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvNew2022Restored3].[Production].[WorkOrderRouting] cannot
be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000004_4A4E069C' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4A4E069C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4A4E069C] on [Adv2022ShalevSoft].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4A4E069C] on [Adv2022ShalevSoft].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [Adv2022ShalevSoft].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [Adv2022ShalevSoft].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME (
S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence]
WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Adv2022ShalevSoft].[Production].[WorkOrderRouting] is updated.' ;

END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Adv2022ShalevSoft].[Production].[WorkOrderRouting] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000004_5E54FF49' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[WorkOrderRouting] [_WA_Sys_00000004_5E54FF49] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_5E54FF49] on [Demo20240411].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_5E54FF49] on [Demo20240411].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
```

```
UPDATE STATISTICS [Demo20240411].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [Demo20240411].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [Demo20240411].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME ( S.OBJECT_ID ) =
'WorkOrderRouting' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence]
WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Demo20240411].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Demo20240411].[Production].[WorkOrderRouting] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvDest20240317].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [AdvDest20240317].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvDest20240317].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [AdvDest20240317].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
```

```
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME (
S.OBJECT_ID )  =  'WorkOrderRouting' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence]
WITH FULLSCAN  ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvDest20240317].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [AdvDest20240317].[Production].[WorkOrderRouting] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_00000004_4A4E069C' AND OBJECT_NAME ( S.OBJECT_ID )  =  'WorkOrderRouting'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4A4E069C] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000004_4A4E069C] on [Advnew2022Moved].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4A4E069C] on [Advnew2022Moved].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'WorkOrderRouting'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN  ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [Advnew2022Moved].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [Advnew2022Moved].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME (
S.OBJECT_ID )  =  'WorkOrderRouting' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence]
WITH FULLSCAN  ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Advnew2022Moved].[Production].[WorkOrderRouting] is updated.' ;
```

```
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [Advnew2022Moved].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000004_4B422AD5' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[WorkOrderRouting] [_WA_Sys_00000004_4B422AD5] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000004_4B422AD5] on [newdbemo3099].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000004_4B422AD5] on [newdbemo3099].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_WorkOrderRouting_ProductID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[WorkOrderRouting] [IX_WorkOrderRouting_ProductID] WITH FULLSCAN ;
END
PRINT '[IX_WorkOrderRouting_ProductID] on [newdbemo3099].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_WorkOrderRouting_ProductID] on [newdbemo3099].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence' AND OBJECT_NAME ( S.OBJECT_ID ) = 'WorkOrderRouting' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Production].[WorkOrderRouting] [PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] WITH FULLSCAN ;
END
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [newdbemo3099].[Production].[WorkOrderRouting] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence] on [newdbemo3099].[Production].[WorkOrderRouting] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
```

```
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated
< = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <
= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [newdbemo3099].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [newdbemo3099].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000D_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5E8A0973] WITH FULLSCAN ;
```

```
END
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000E_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000F_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
```

```
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000013_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5E8A0973] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Advnew2022Moved].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Advnew2022Moved].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
```

```
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
```

```
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
```

```
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvDest20240317].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvDest20240317].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000D_72910220' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_72910220] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
```

```
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000E_72910220' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_72910220] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000F_72910220' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_72910220] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000010_72910220' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_72910220] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000011_72910220' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_72910220] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000013_72910220' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_72910220] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_72910220] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
```

```
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated
< = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <
= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Demo20240411].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Demo20240411].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000D_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000E_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
```

```
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000F_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
```

```
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_00000013_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5E8A0973] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000013_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5E8A0973] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID )  =
'SalesOrderHeader' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN  ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
```

```
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
```

```
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
```

```
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
```

```
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
```

```
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
```

```
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNew2022Restored].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN ;
END
```

```
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
```

```
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P
WHERE S.NAME  =  'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P
WHERE S.NAME  =  'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN  ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P
WHERE S.NAME  =  'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND P.last_updated  <
=  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN  ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AIDBAADV2' )
```

```
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P
WHERE S.NAME  =  'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN  ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AIDBAADV2].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AIDBAADV2].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader'
AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
```

```
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
```

```
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = '_WA_Sys_0000000D_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = '_WA_Sys_0000000E_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = '_WA_Sys_0000000F_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
```

```
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  '_WA_Sys_00000010_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5E8A0973] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000010_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  '_WA_Sys_00000011_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5E8A0973] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000011_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  '_WA_Sys_00000013_5E8A0973' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5E8A0973] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000013_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5E8A0973] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeader' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN  ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Test33].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
```

```
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Test33].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000D_5E8A0973'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000E_5E8A0973'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_0000000F_5E8A0973'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000010_5E8A0973'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
```

```
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000010_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000011_5E8A0973'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000013_5E8A0973'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5E8A0973] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5E8A0973] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
```

```
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME =
'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN
;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME =
'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000D_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [_WA_Sys_0000000D_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000D_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000E_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [_WA_Sys_0000000E_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000E_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_0000000F_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [_WA_Sys_0000000F_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_0000000F_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000010_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [_WA_Sys_00000010_5F7E2DAC] WITH FULLSCAN ;
```

```
END
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000010_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000011_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [_WA_Sys_00000011_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000011_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000013_5F7E2DAC' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [_WA_Sys_00000013_5F7E2DAC] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000013_5F7E2DAC] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_SalesOrderHeader_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_rowguid] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_rowguid] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
```

```
P WHERE S.NAME = 'AK_SalesOrderHeader_SalesOrderNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [AK_SalesOrderHeader_SalesOrderNumber] WITH FULLSCAN ;
END
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_SalesOrderHeader_SalesOrderNumber] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderHeader_CustomerID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_CustomerID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_CustomerID] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_SalesOrderHeader_SalesPersonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated
< = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [IX_SalesOrderHeader_SalesPersonID] WITH FULLSCAN ;
END
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_SalesOrderHeader_SalesPersonID] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderHeader_SalesOrderID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeader' AND P.last_updated <
= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeader] [PK_SalesOrderHeader_SalesOrderID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdventureWorks].[Sales].[SalesOrderHeader] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeader_SalesOrderID] on [AdventureWorks].[Sales].[SalesOrderHeader] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdventureWorks].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdventureWorks].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdventureWorks].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdventureWorks].[Sales].[SalesOrderHeaderSalesReason] cannot
be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000002_6DCC4D03'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) )
)
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6DCC4D03] WITH
FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Advnew2022_20240312102058].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME =
'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
```

```
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN  ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on
[Advnew2022_20240312102058].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on
[Advnew2022_20240312102058].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  '_WA_Sys_00000002_6DCC4D03' AND OBJECT_NAME ( S.OBJECT_ID )  =  'SalesOrderHeaderSalesReason' AND P.last_updated  <
=  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6DCC4D03] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Test33].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Test33].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME ( S.OBJECT_ID )  =
'SalesOrderHeaderSalesReason' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Sales].[SalesOrderHeaderSalesReason] [PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH
FULLSCAN  ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Test33].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Test33].[Sales].[SalesOrderHeaderSalesReason] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID )  =
'SalesOrderHeaderSalesReason' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNewDB2022Portal].[Sales].[SalesOrderHeaderSalesReason]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [AIDBAADV2].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6EC0713C] on [AIDBAADV2].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Sales].[SalesOrderHeaderSalesReason] [PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID]
WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AIDBAADV2].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AIDBAADV2].[Sales].[SalesOrderHeaderSalesReason] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID ) =
```

```
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNew2022Restored].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNew2022Restored].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNew2022Restored].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNew2022Restored].[Sales].[SalesOrderHeaderSalesReason]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNew2022Restored2].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNew2022Restored2].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNew2022Restored2].[Sales].[SalesOrderHeaderSalesReason]
```

```
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNew2022Restored3].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvNew2022Restored3].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvNew2022Restored3].[Sales].[SalesOrderHeaderSalesReason]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000002_6DCC4D03' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6DCC4D03] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME (
```

```
S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Adv2022ShalevSoft].[Sales].[SalesOrderHeaderSalesReason]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000002_01D345B0' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_01D345B0] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_01D345B0] on [Demo20240411].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_01D345B0] on [Demo20240411].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Demo20240411].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Demo20240411].[Sales].[SalesOrderHeaderSalesReason] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvDest20240317].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[_WA_Sys_00000002_6EC0713C] on [AdvDest20240317].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvDest20240317].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [AdvDest20240317].[Sales].[SalesOrderHeaderSalesReason] cannot
be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000002_6DCC4D03' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6DCC4D03] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Advnew2022Moved].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6DCC4D03] on [Advnew2022Moved].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Advnew2022Moved].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [Advnew2022Moved].[Sales].[SalesOrderHeaderSalesReason]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
```

```
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000002_6EC0713C' AND OBJECT_NAME ( S.OBJECT_ID ) = 'SalesOrderHeaderSalesReason' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeaderSalesReason] [_WA_Sys_00000002_6EC0713C] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000002_6EC0713C] on [newdbemo3099].[Sales].[SalesOrderHeaderSalesReason] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000002_6EC0713C] on [newdbemo3099].[Sales].[SalesOrderHeaderSalesReason] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'SalesOrderHeaderSalesReason' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Sales].[SalesOrderHeaderSalesReason]
[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] WITH FULLSCAN ;
END
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [newdbemo3099].[Sales].[SalesOrderHeaderSalesReason] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID] on [newdbemo3099].[Sales].[SalesOrderHeaderSalesReason] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [Advnew2022Moved].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [Advnew2022Moved].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022Moved' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022Moved].sys.stats S CROSS APPLY [Advnew2022Moved].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022Moved].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Advnew2022Moved].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Advnew2022Moved].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [newdbemo3099].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [newdbemo3099].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'newdbemo3099' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [newdbemo3099].sys.stats S CROSS APPLY [newdbemo3099].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [newdbemo3099].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [newdbemo3099].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [newdbemo3099].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [Demo20240411].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [Demo20240411].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
```

```
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Demo20240411].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Demo20240411].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [AdvDest20240317].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [AdvDest20240317].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvDest20240317].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvDest20240317].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [AdvNew2022Restored3].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [AdvNew2022Restored3].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
```

```
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNew2022Restored3].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNew2022Restored3].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [Adv2022ShalevSoft].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [Adv2022ShalevSoft].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Adv2022ShalevSoft].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Adv2022ShalevSoft].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [AdvNew2022Restored].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [AdvNew2022Restored].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNew2022Restored].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNew2022Restored].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [AdvNew2022Restored2].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [AdvNew2022Restored2].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNew2022Restored2].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNew2022Restored2].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [AdvNewDB2022Portal].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [AdvNewDB2022Portal].[Person].[BusinessEntity] cannot be updated.' ;
```

```
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNewDB2022Portal].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdvNewDB2022Portal].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [AIDBAADV2].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [AIDBAADV2].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AIDBAADV2].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AIDBAADV2].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
```

```
PRINT '[AK_BusinessEntity_rowguid] on [Advnew2022_20240312102058].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [Advnew2022_20240312102058].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID'
AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Advnew2022_20240312102058].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Advnew2022_20240312102058].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [Test33].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [Test33].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Test33].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [Test33].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_BusinessEntity_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < =
```

```
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[BusinessEntity] [AK_BusinessEntity_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_BusinessEntity_rowguid] on [AdventureWorks].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_BusinessEntity_rowguid] on [AdventureWorks].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_BusinessEntity_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'BusinessEntity' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[BusinessEntity] [PK_BusinessEntity_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdventureWorks].[Person].[BusinessEntity] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_BusinessEntity_BusinessEntityID] on [AdventureWorks].[Person].[BusinessEntity] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [AdventureWorks].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [AdventureWorks].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdventureWorks].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdventureWorks].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
```

```
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND P.last_updated < = DATEADD
( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [AdventureWorks].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [AdventureWorks].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [AdventureWorks].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [AdventureWorks].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdventureWorks].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdventureWorks].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [AdventureWorks].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [AdventureWorks].[Person].[Person] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000003_05D8E0BE' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[PersonPhone] [_WA_Sys_00000003_05D8E0BE] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdventureWorks].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdventureWorks].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [AdventureWorks].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [AdventureWorks].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdventureWorks' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdventureWorks].sys.stats S CROSS APPLY [AdventureWorks].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'PersonPhone' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdventureWorks].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] WITH
FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdventureWorks].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdventureWorks].[Person].[PersonPhone] cannot be updated.'
;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
```

```
END
PRINT '[IX_EmailAddress_EmailAddress] on [Advnew2022_20240312102058].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [Advnew2022_20240312102058].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME =
'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH
FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Advnew2022_20240312102058].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Advnew2022_20240312102058].[Person].[EmailAddress] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [Advnew2022_20240312102058].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [Advnew2022_20240312102058].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_Person_rowguid' AND
OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [Advnew2022_20240312102058].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [Advnew2022_20240312102058].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
```

```
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =
'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID )  =  'Person' AND P.last_updated  < =  DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN  ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Advnew2022_20240312102058].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Advnew2022_20240312102058].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_Person_BusinessEntityID' AND
OBJECT_NAME ( S.OBJECT_ID )  =  'Person' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN  ;
END
PRINT '[PK_Person_BusinessEntityID] on [Advnew2022_20240312102058].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [Advnew2022_20240312102058].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_00000003_04E4BC85'
AND OBJECT_NAME ( S.OBJECT_ID )  =  'PersonPhone' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[PersonPhone] [_WA_Sys_00000003_04E4BC85] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000003_04E4BC85] on [Advnew2022_20240312102058].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_04E4BC85] on [Advnew2022_20240312102058].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE S.NAME  =  'IX_PersonPhone_PhoneNumber'
AND OBJECT_NAME ( S.OBJECT_ID )  =  'PersonPhone' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN  ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [Advnew2022_20240312102058].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [Advnew2022_20240312102058].[Person].[PersonPhone] cannot be updated.' ;
```

```
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Advnew2022_20240312102058' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Advnew2022_20240312102058].sys.stats S CROSS APPLY
[Advnew2022_20240312102058].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE S.NAME =
'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Advnew2022_20240312102058].[Person].[PersonPhone]
[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] WITH FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Advnew2022_20240312102058].[Person].[PersonPhone] is
updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Advnew2022_20240312102058].[Person].[PersonPhone]
cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [AIDBAADV2].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [AIDBAADV2].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AIDBAADV2].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AIDBAADV2].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND P.last_updated < = DATEADD (
```

```
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [AIDBAADV2].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [AIDBAADV2].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [AIDBAADV2].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [AIDBAADV2].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AIDBAADV2].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AIDBAADV2].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P
WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [AIDBAADV2].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [AIDBAADV2].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AIDBAADV2' )
```

```
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P
WHERE S.NAME  =  '_WA_Sys_00000003_05D8E0BE' AND OBJECT_NAME ( S.OBJECT_ID )  =  'PersonPhone' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[PersonPhone] [_WA_Sys_00000003_05D8E0BE] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AIDBAADV2].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AIDBAADV2].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P
WHERE S.NAME  =  'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID )  =  'PersonPhone' AND P.last_updated  < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN  ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [AIDBAADV2].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [AIDBAADV2].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AIDBAADV2' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AIDBAADV2].sys.stats S CROSS APPLY [AIDBAADV2].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P
WHERE S.NAME  =  'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME ( S.OBJECT_ID )  =
'PersonPhone' AND P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AIDBAADV2].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] WITH
FULLSCAN  ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AIDBAADV2].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AIDBAADV2].[Person].[PersonPhone] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'Test33' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id )  P WHERE
S.NAME  =  'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID )  =  'EmailAddress' AND P.last_updated  < =  DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN  ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [Test33].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
```

```
PRINT '[IX_EmailAddress_EmailAddress] on [Test33].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND P.last_updated <
= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Test33].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Test33].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [Test33].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [Test33].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [Test33].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [Test33].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
```

```
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Test33].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Test33].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [Test33].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [Test33].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = '_WA_Sys_00000003_04E4BC85' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[PersonPhone] [_WA_Sys_00000003_04E4BC85] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_04E4BC85] on [Test33].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_04E4BC85] on [Test33].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [Test33].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [Test33].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Test33' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Test33].sys.stats S CROSS APPLY [Test33].sys.dm_db_stats_properties ( S.object_id,S.stats_id ) P WHERE
```

```
S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Test33].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] WITH
FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Test33].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Test33].[Person].[PersonPhone] cannot be updated.' ;

END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNewDB2022Portal].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNewDB2022Portal].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'EmailAddress' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNewDB2022Portal].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNewDB2022Portal].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [AdvNewDB2022Portal].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [AdvNewDB2022Portal].[Person].[Password] cannot be updated.' ;
END CATCH ;
END
```

```
IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [AdvNewDB2022Portal].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [AdvNewDB2022Portal].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNewDB2022Portal].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNewDB2022Portal].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [AdvNewDB2022Portal].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [AdvNewDB2022Portal].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000003_05D8E0BE' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[PersonPhone] [_WA_Sys_00000003_05D8E0BE] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNewDB2022Portal].[Person].[PersonPhone] is updated.' ;
END TRY
```

```
BEGIN CATCH
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNewDB2022Portal].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNewDB2022Portal].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNewDB2022Portal].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNewDB2022Portal' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNewDB2022Portal].sys.stats S CROSS APPLY [AdvNewDB2022Portal].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNewDB2022Portal].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID]
WITH FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNewDB2022Portal].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNewDB2022Portal].[Person].[PersonPhone] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNew2022Restored].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNew2022Restored].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) =
```

```
'EmailAddress' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNew2022Restored].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNew2022Restored].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [AdvNew2022Restored].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [AdvNew2022Restored].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [AdvNew2022Restored].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [AdvNew2022Restored].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNew2022Restored].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNew2022Restored].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
```

```
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [AdvNew2022Restored].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [AdvNew2022Restored].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000003_05D8E0BE' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[PersonPhone] [_WA_Sys_00000003_05D8E0BE] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNew2022Restored].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNew2022Restored].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNew2022Restored].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNew2022Restored].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored].sys.stats S CROSS APPLY [AdvNew2022Restored].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID]
WITH FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNew2022Restored].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNew2022Restored].[Person].[PersonPhone] cannot be
```

```
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNew2022Restored2].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNew2022Restored2].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'EmailAddress' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNew2022Restored2].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNew2022Restored2].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [AdvNew2022Restored2].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [AdvNew2022Restored2].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
```

```
END
PRINT '[AK_Person_rowguid] on [AdvNew2022Restored2].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [AdvNew2022Restored2].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNew2022Restored2].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNew2022Restored2].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [AdvNew2022Restored2].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [AdvNew2022Restored2].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000003_05D8E0BE' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[PersonPhone] [_WA_Sys_00000003_05D8E0BE] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNew2022Restored2].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNew2022Restored2].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
```

```
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNew2022Restored2].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNew2022Restored2].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored2' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored2].sys.stats S CROSS APPLY [AdvNew2022Restored2].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored2].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID]
WITH FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNew2022Restored2].[Person].[PersonPhone] is updated.'
;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNew2022Restored2].[Person].[PersonPhone] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [Adv2022ShalevSoft].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [Adv2022ShalevSoft].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'EmailAddress' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Adv2022ShalevSoft].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Adv2022ShalevSoft].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
```

```
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [Adv2022ShalevSoft].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [Adv2022ShalevSoft].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [Adv2022ShalevSoft].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [Adv2022ShalevSoft].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Adv2022ShalevSoft].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Adv2022ShalevSoft].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [Adv2022ShalevSoft].[Person].[Person] is updated.' ;
```

```
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [Adv2022ShalevSoft].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000003_04E4BC85' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[PersonPhone] [_WA_Sys_00000003_04E4BC85] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_04E4BC85] on [Adv2022ShalevSoft].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_04E4BC85] on [Adv2022ShalevSoft].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [Adv2022ShalevSoft].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [Adv2022ShalevSoft].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Adv2022ShalevSoft' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Adv2022ShalevSoft].sys.stats S CROSS APPLY [Adv2022ShalevSoft].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Adv2022ShalevSoft].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID]
WITH FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Adv2022ShalevSoft].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Adv2022ShalevSoft].[Person].[PersonPhone] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
```

```
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNew2022Restored3].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [AdvNew2022Restored3].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'EmailAddress' AND P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNew2022Restored3].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvNew2022Restored3].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND
P.last_updated <= DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [AdvNew2022Restored3].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [AdvNew2022Restored3].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated <=
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [AdvNew2022Restored3].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [AdvNew2022Restored3].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
```

```
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNew2022Restored3].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvNew2022Restored3].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [AdvNew2022Restored3].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [AdvNew2022Restored3].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = '_WA_Sys_00000003_05D8E0BE' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[PersonPhone] [_WA_Sys_00000003_05D8E0BE] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNew2022Restored3].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvNew2022Restored3].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNew2022Restored3].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [AdvNew2022Restored3].[Person].[PersonPhone] cannot be updated.' ;
```

```
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvNew2022Restored3' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvNew2022Restored3].sys.stats S CROSS APPLY [AdvNew2022Restored3].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME (
S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvNew2022Restored3].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID]
WITH FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNew2022Restored3].[Person].[PersonPhone] is updated.'
;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [AdvNew2022Restored3].[Person].[PersonPhone] cannot be
updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [Demo20240411].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [Demo20240411].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Demo20240411].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [Demo20240411].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND P.last_updated < = DATEADD
( HOUR,-48,SYSDATETIME ( ) ) )
```

```
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [Demo20240411].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [Demo20240411].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD ( HOUR,-
48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [Demo20240411].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [Demo20240411].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Demo20240411].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [Demo20240411].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < = DATEADD (
HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Person_BusinessEntityID] on [Demo20240411].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [Demo20240411].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
```

```
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = '_WA_Sys_00000003_18EBB532' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[PersonPhone] [_WA_Sys_00000003_18EBB532] WITH FULLSCAN ;
END
PRINT '[_WA_Sys_00000003_18EBB532] on [Demo20240411].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_18EBB532] on [Demo20240411].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'IX_PersonPhone_PhoneNumber' AND OBJECT_NAME ( S.OBJECT_ID ) = 'PersonPhone' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[PersonPhone] [IX_PersonPhone_PhoneNumber] WITH FULLSCAN ;
END
PRINT '[IX_PersonPhone_PhoneNumber] on [Demo20240411].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_PersonPhone_PhoneNumber] on [Demo20240411].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'Demo20240411' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [Demo20240411].sys.stats S CROSS APPLY [Demo20240411].sys.dm_db_stats_properties ( S.object_id,S.stats_id )
P WHERE S.NAME = 'PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'PersonPhone' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [Demo20240411].[Person].[PersonPhone] [PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] WITH
FULLSCAN ;
END
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Demo20240411].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID] on [Demo20240411].[Person].[PersonPhone] cannot be updated.'
;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_EmailAddress_EmailAddress' AND OBJECT_NAME ( S.OBJECT_ID ) = 'EmailAddress' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[EmailAddress] [IX_EmailAddress_EmailAddress] WITH FULLSCAN ;
END
PRINT '[IX_EmailAddress_EmailAddress] on [AdvDest20240317].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_EmailAddress_EmailAddress] on [AdvDest20240317].[Person].[EmailAddress] cannot be updated.' ;
```

```
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_EmailAddress_BusinessEntityID_EmailAddressID' AND OBJECT_NAME ( S.OBJECT_ID ) =
'EmailAddress' AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[EmailAddress] [PK_EmailAddress_BusinessEntityID_EmailAddressID] WITH FULLSCAN ;
END
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvDest20240317].[Person].[EmailAddress] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_EmailAddress_BusinessEntityID_EmailAddressID] on [AdvDest20240317].[Person].[EmailAddress] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'PK_Password_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Password' AND
P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[Password] [PK_Password_BusinessEntityID] WITH FULLSCAN ;
END
PRINT '[PK_Password_BusinessEntityID] on [AdvDest20240317].[Person].[Password] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Password_BusinessEntityID] on [AdvDest20240317].[Person].[Password] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'AK_Person_rowguid' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person' AND P.last_updated < =
DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[Person] [AK_Person_rowguid] WITH FULLSCAN ;
END
PRINT '[AK_Person_rowguid] on [AdvDest20240317].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[AK_Person_rowguid] on [AdvDest20240317].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state = 0 AND name = 'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id ) P WHERE S.NAME = 'IX_Person_LastName_FirstName_MiddleName' AND OBJECT_NAME ( S.OBJECT_ID ) = 'Person'
AND P.last_updated < = DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[Person] [IX_Person_LastName_FirstName_MiddleName] WITH FULLSCAN ;
END
```

```
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvDest20240317].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[IX_Person_LastName_FirstName_MiddleName] on [AdvDest20240317].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  'PK_Person_BusinessEntityID' AND OBJECT_NAME ( S.OBJECT_ID )  =  'Person' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[Person] [PK_Person_BusinessEntityID] WITH FULLSCAN  ;
END
PRINT '[PK_Person_BusinessEntityID] on [AdvDest20240317].[Person].[Person] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[PK_Person_BusinessEntityID] on [AdvDest20240317].[Person].[Person] cannot be updated.' ;
END CATCH ;
END

IF EXISTS
( SELECT * FROM sys.databases WHERE state  =  0 AND name  =  'AdvDest20240317' )
BEGIN
BEGIN TRY
IF EXISTS  ( SELECT 1 FROM [AdvDest20240317].sys.stats S CROSS APPLY [AdvDest20240317].sys.dm_db_stats_properties (
S.object_id,S.stats_id )  P WHERE S.NAME  =  '_WA_Sys_00000003_05D8E0BE' AND OBJECT_NAME ( S.OBJECT_ID )  =  'PersonPhone' AND
P.last_updated  < =  DATEADD ( HOUR,-48,SYSDATETIME ( ) ) )
BEGIN
UPDATE STATISTICS [AdvDest20240317].[Person].[PersonPhone] [_WA_Sys_00000003_05D8E0BE] WITH FULLSCAN  ;
END
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvDest20240317].[Person].[PersonPhone] is updated.' ;
END TRY
BEGIN CATCH
PRINT '[_WA_Sys_00000003_05D8E0BE] on [AdvDest20240317].[Person].[PersonPhone] cannot be updated.' ;
END CATCH ;
END
```
