# Data Migration Best Practices

## A How-To Guide for Data Teams

Datafold

# A How-To Guide For Data Teams

As most companies grow and evolve, they need new systems to support the increased size and complexity of their data. Once executive sponsorship exists, it paves the way for a database migration, and data engineers face a series of critical decisions that define project execution. They must choose the right technology for the job, determine their strategy for how to move the data and code, and decide whether to outsource or handle the migration internally, among other decisions. A successful migration occurs when data in the new system and the legacy's match, and most importantly, the business fully trusts and can verify that parity.

In this guide, we share the lessons we've learned from large-scale database migrations at fast-growing companies like **Lyft**, **Faire**, and **Eventbrite**. We first define the objectives, success factors, and metrics for a database migration. Next, we share six best practices to keep in mind when tackling one at your company. Finally, we present a four-step process for getting the job done.

# Table Of Contents

# In The Spotlight

A ride whenever you need one

Tables: **500**

Team Size: **200**

Redshift → Hive

FAIRE

The online wholesale marketplace for retailers

Tables: **5,000**

Team Size: **115**

Redshift → Snowflake

eventbrite

The global event discovery and ticketing platform

Tables: **300**

Team Size: **50**

Presto → Snowflake

# What Is A Data Migration?

Data migrations are often complex and challenging, frequently exceeding time and budget expectations. Nevertheless, they're an essential step for companies that have outgrown their existing systems.

The term "database migration" refers to moving data, and/or the transformation code that supports it, from one data warehouse/lakehouse to another. Faire, for example, recently shifted its data from Amazon Redshift to Snowflake. Common data migrations undergone by data teams are from legacy on-premise databases to cloud databases like Google BigQuery and Snowflake or from legacy transformation frameworks (e.g. Alteryx, Informatic, or stored procedures) to modern ones like dbt or Dagster.

If your current setup doesn't support your anticipated scale, the ability to integrate with other tools is limited, or the operational costs have become unsustainable it may be time to migrate to a more scalable solution.

The journey of migration is undoubtedly challenging. In this guide, we share key lessons learned from helping Lyft, Faire, and Eventbrite migrate their large-scale databases. Our hope is that it will save you time, energy, and money while navigating this process.

# Top Reasons For Data Migrations

Organizations undergo data migrations for a variety of reasons: it may be part of a company-wide initiative for modernization, the emergence of better technology in the market, or the inadequacy of existing platforms to stakeholder needs.

Whatever the specific reason, the cause for a migration typically falls under one of three categories: concerns of scalability, cost, and interoperability.

1. **Scalability**—the need to support more data or greater performance demands. This pain is likely felt across the organization: BI tools users saying "the tool is too slow" or data engineers unable to differentiate one stored procedure from another. A data migration means adopting practices and technologies that allow for greater optimization, improved performance, and better governance—all of which are aimed at helping organizations scale.

2. **Cost**—the need to lower overall storage and compute costs. When Redshift and Snowflake introduced cloud computing with cheap storage options, the data world was changed forever. As a result, the data community has seen this mass migration of the adoption of Modern Data Stack (MDS) technologies—built on cloud computing and modern transformation technologies like dbt. Compared to on-premise database solutions that are expensive to maintain and scale, many data teams are migrating to save on long-term storage costs.

3. **Interoperability**—the need to support integrations with other systems. A data migration in 2023 probably looks considerably different than it did in 2016. In 2016, it might have just been migrating from Oracle to Snowflake. In 2023, a data migration will likely still include a similar database migration, but with some additions on top: new transformation workflow, ETL tools, new

BI tool, and the adoption of other MDS tooling like reverse ETL. These MDS technologies offer many integrations with each other, making it easier than ever to ensure interoperability between your tooling choices, that your legacy systems may not have had or supported.

# 3 Objectives For Data Migration Success

**1. Adoption by your stakeholders**

A migration is only successful when your stakeholders (business users, data science and machine learning teams, etc.) fully adopt the new platform and stop relying on (and querying) the old platform.

> Track adoption by calculating the percentage of tables and data applications (e.g. BI reports) succesfully migrated over.

**2. Parity between systems**

Parity is the necessary prerequisite for adoption. For data-driven businesses, users are highly sensitive to data quality. During a data migration, the critical aspect of data quality is parity. FP&A and C-Suite leadership will not tolerate historical metrics shifting due to a migration, and ML teams need to ensure there is no loss of data integrity to impact their models.

> Measure parity through data diffs: compute the difference between any two data assets in the legacy and new systems.

**3. Timeline & Cost**

Migrations typically incur high costs:

- **Labor**: Significant data engineering work is required for migrating each data asset, whether outsourced or in-house.

- **Legacy stack**: Until the new stack achieves parity and is fully adopted, the old stack must remain operational, potentially doubling data infrastructure costs for a considerable duration (months to years).

- **Opportunity cost**: Often overlooked, this is the cost of not building essential new data products for the business while resources are diverted to a migration. For businesses that leverage data as a competitive advantage or have leaner data teams, this can be substantial.

To mitigate migration costs, we share our best practices below.

# 6 Best Practices For Effective Data Migrations

Effective database migrations require the right mix of people, processes, and technology. By adopting the following best practices, you can improve your chances of delivering your database migration on time and budget with maximum acceptance from key stakeholders.

## 1. Be comprehensive in evaluating technology vendors

Before you commit to a large-scale database migration, you'll want to identify the right technology for your business. To prevent critical errors, it's essential that you evaluate each technology vendor against your unique requirements upfront. The last thing you want is to end up with more problems than your team experiences today.

We recommend including five criteria in your evaluation:

- **Scalability** — Is this system capable of supporting your business as it grows over the next 5-10 years?
- **User experience** — Is this system intuitive for non-technical business stakeholders to use? Does it empower them to be productive?
- **Interoperability** — Is this system open and connected? Does it easily integrate with other tools within your tech stack?
- **Trajectory** — Is this system neither immature nor so legacy that it will become obsolete in a few years?
- **Cost** — Is this system affordable today? Will it be cost-effective as the business scales?

### ⚠️ Beware Of The Open-Source Trap ⚠️

If you're motivated to reduce costs alone, beware that it can be very difficult to forecast what you'll spend in reality. This is especially true if you're leveraging a data lakehouse or open-source database. In the latter case, be sure to take the

cost of maintenance into account. Many choose open-source to avoid locked-in contracts, but don't account for the associated costs and level of difficulty. These projects eat up (costly) engineering hours!

> "When picking the technology stack to migrate to at Lyft, we underestimated the complexity and cost of hosting a distributed open-source engine like Hive. We also initially overshot the lifecycle of the chosen technology: by the time we fully rolled out Hive 3 years into the start of the migration project – which at the start of the project was at the peak of its popularity – Hive was already considered legacy and no longer met business needs forcing us to accelerate the rollout of Spark and Trino which, in turn, required a sub-migration from Hive to those engines."

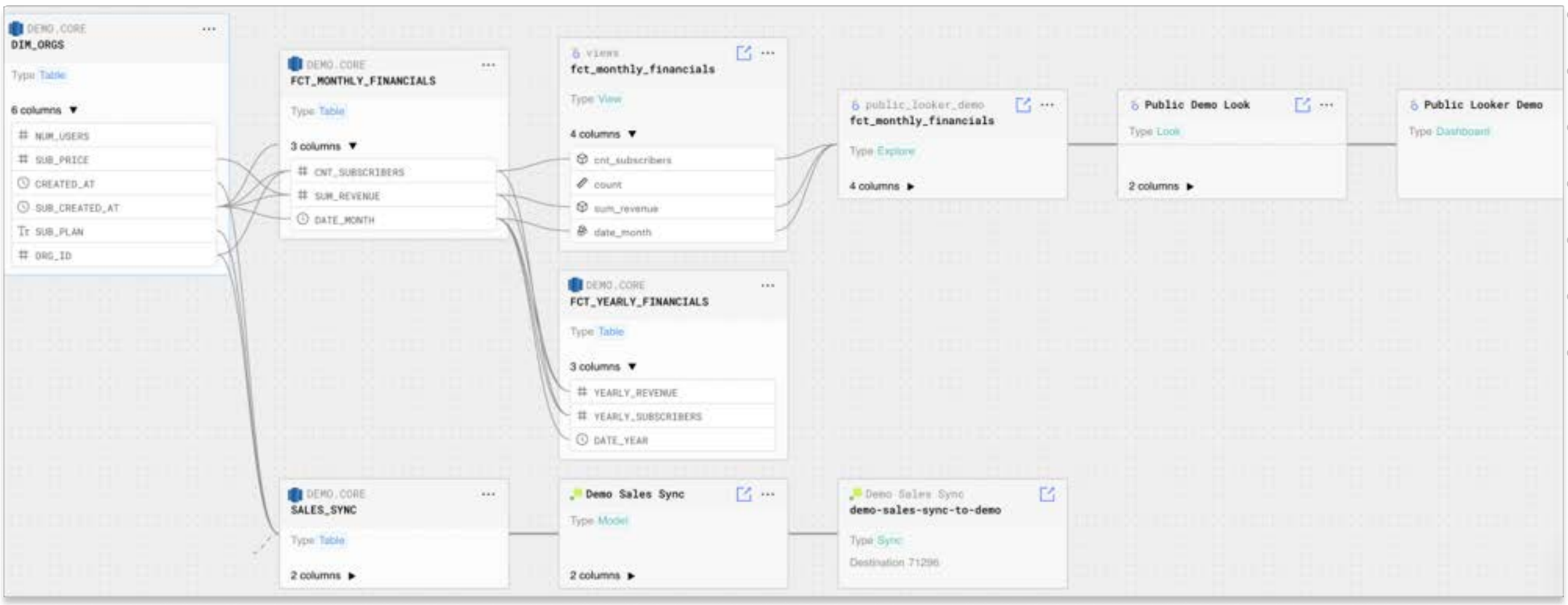
**Faire** & **Eventbrite** chose modern hosted cloud data warehouses, opting in for their fully managed service and turnkey performance and features at the expense of flexibility and vendor lock-in.

## 2. Plan and prioritize asset migration appropriately

Planning ahead ensures a smooth migration process. We highlight two critical considerations: the importance of mapping and prioritizing data assets, and why it's better to migrate data consumption endpoints before data production pipelines.

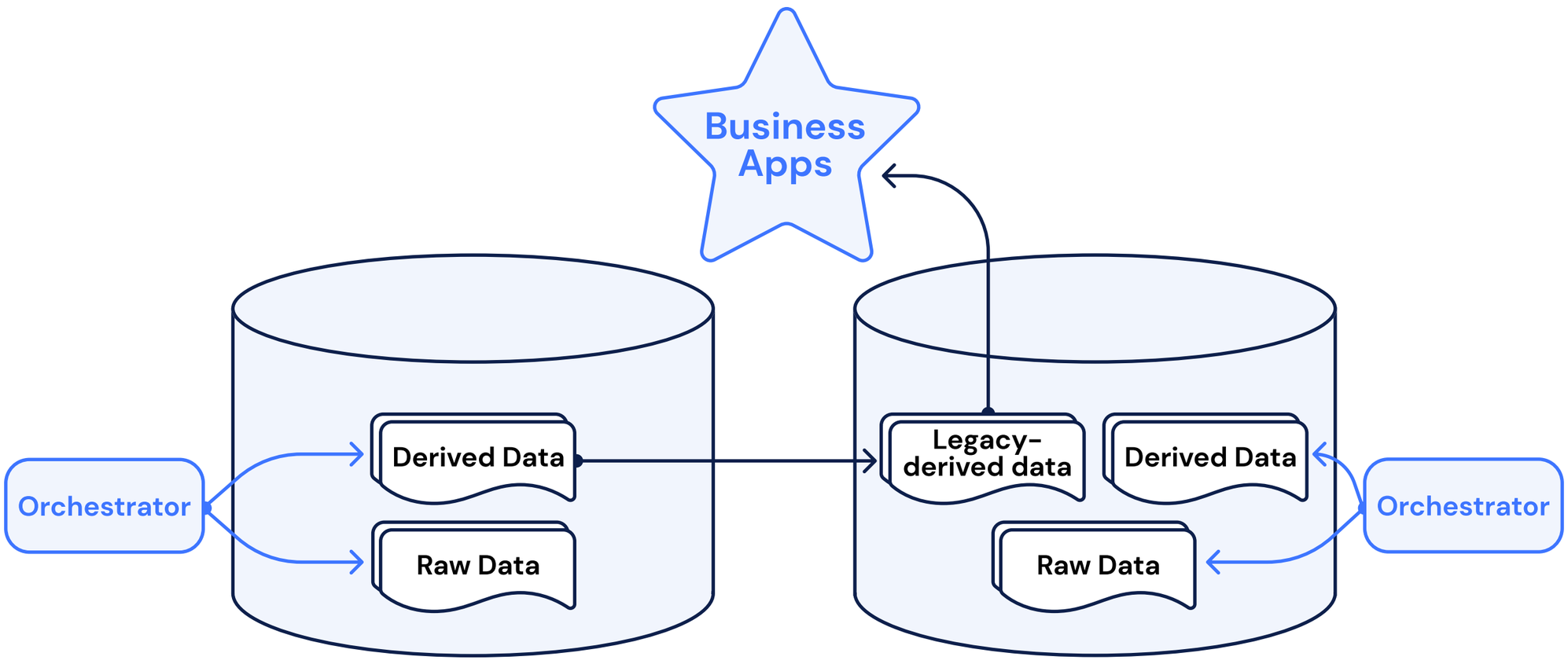
***Identify assets***

You should first leverage column-level lineage to identify which assets are most vital to the organization and rank them in order of priority. This can help clarify how the business uses each asset, what dependencies exist, and how important each table, column, and dashboard is. Using column-level lineage, you can more easily identify important tables (i.e. which tables having hundreds of important downstream nodes?) or assets you no longer need to migrate because of a lack of use or downstream nodes. This will allow your team to approach the migration in bite-sized phases.




***Migrate data consumption before data production***

A mature data platform has a lot of production pipelines ingesting and transforming data, as well as numerous data consumption endpoints, such as BI reports, dashboards, and data exports.




When planning the migration, consider moving over the consumption to the new system/database first. For example, you can replicate the data produced in the old database to the new database, and let data users and apps query it from the new system. That achieves three important goals:

1. Reduce the load on the old system, which is likely under immense pressure already
2. Provide your team more leeway in planning and executing further migration, since all of the data users are working with the new system.
3. You can run data diffs in the new system, comparing the data produced by the old system with the data produced natively on the new systems to speed up user acceptance

## 3. Lift and shift the data in its current state

When architecting the migration, you'll need to choose between two approaches: You can either:

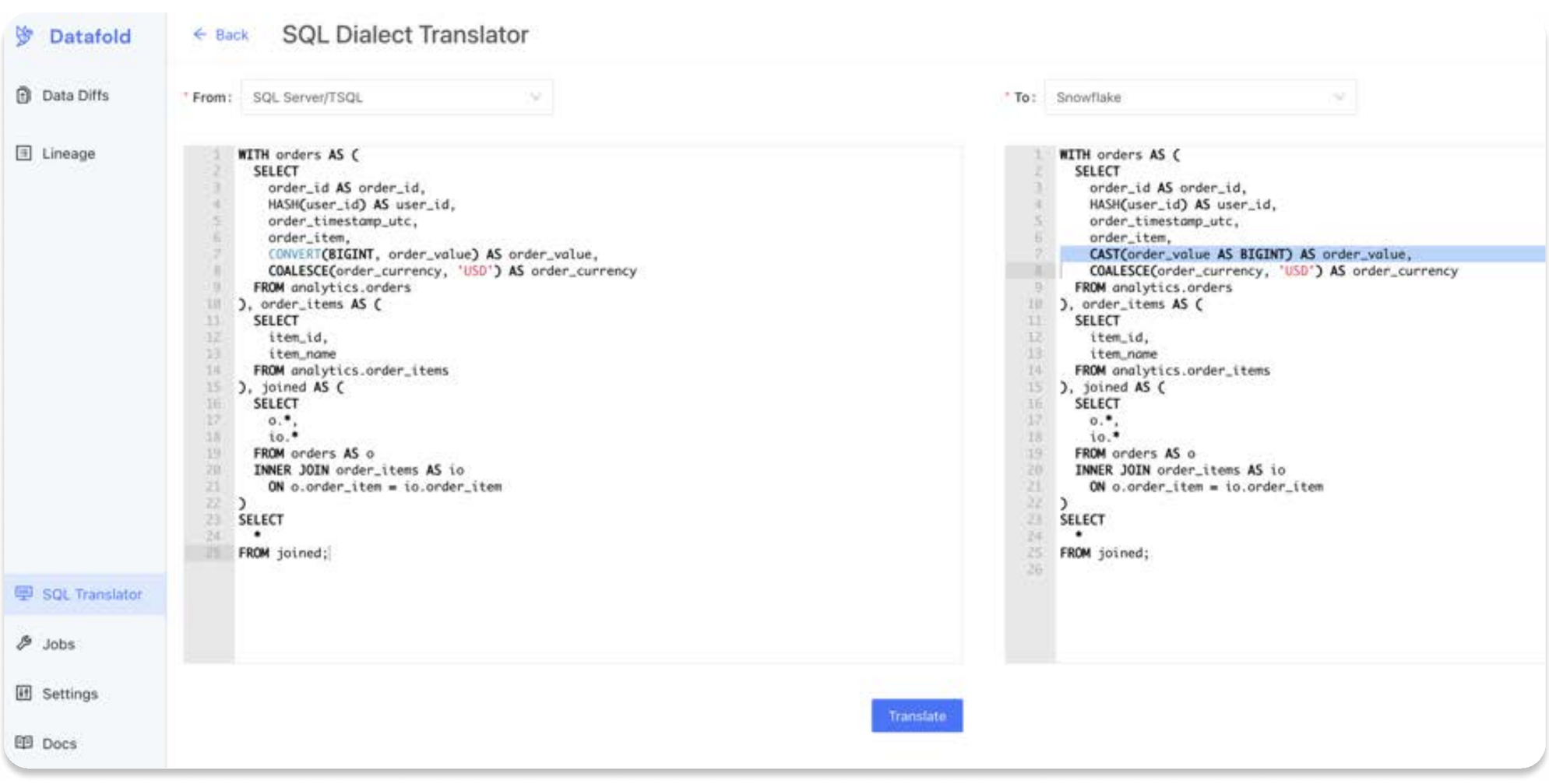
A) Take everything that lives in the old database and port it into the new database with minimal modifications (e.g., you might still deprecate unused data assets),

B) Remodel the data and refactor the code in the process to achieve your ideal state. Many data engineers find themselves tempted to remodel. But, for many organizations, this is a mistake. Attempting to remodel the data and refactor the code while migrating your database adds considerable complexity to the project, which not only comes with a higher risk of failure but also demands a larger investment of resources. When Datafold CEO Mezhanskiy worked at Lyft, his team took this approach:

> "As a data engineer at Lyft responsible for the migration strategy, I made the grave mistake of embarking on rearchitecting and remodeling the data during the migration. It was very tempting: the old data model was far from elegant and had lots of tech debt. However, having to remodel, rebuild and, worst of all, get buy-in from stakeholders to adopt the new data model added not weeks or months but years to the overall project."

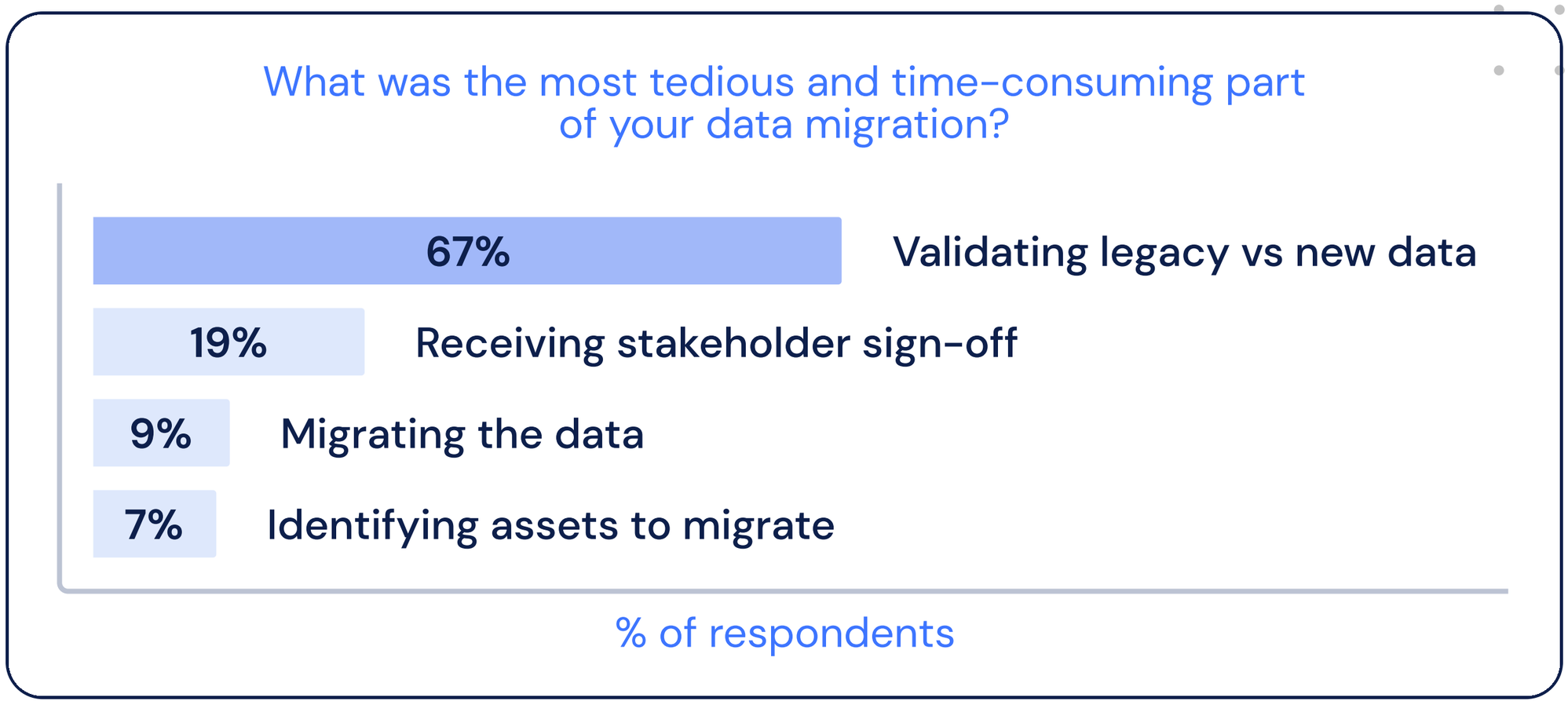
Faire, in contrast, took the lift-and-shift approach and completed the migration with six months to spare.

**Bonus points!** If you're opting to lift-and-shift, we recommend using a SQL translator, like the one supported in Datafold Cloud, to help lift and shift code over from legacy SQL dialect to the new one. Datafold's built-in SQL Dialect Translator automates the SQL conversion process, allowing teams to rapidly migrate large volumes of code without the need for manual rewrites. This not only speeds up the migration process, but also ensures that the business logic embedded in the SQL code is preserved accurately in the new system, saving significant time (no more Googling "`date_trunc` in BigQuery") and reducing errors.




## 4. Document your strategy and action plan up front

Documenting your migration strategy and action plan brings clarity to your ideas, helping identify and rectify flaws, such as overlooked edge cases. It also facilitates stakeholder buy-in. Detailed, organized plans foster transparency and trust, crucial for gaining approval, especially since 19% of respondents in our survey identified stakeholder sign-off as the biggest challenge in data migrations. Lack of agreement from data consumers can lead to increased costs and data quality issues.




To mitigate this, involve stakeholders early, both hands-on data users and high-level sponsors. Clearly communicate the migration's benefits and your data validation plans, maintaining trust through regular progress updates.

A documented plan also unifies the team, reducing miscommunications and potential delays. For example, a simplified migration action plan includes:

1. Set up a copy of data from the legacy system to the new system  to serve BI and other data consumption use cases from the new stack
2. Migrate the core business models  first as they are essential to the rest of the data model
3. Migrate marketing models next as they have the biggest performance bottlenecks on the old stack
4. Deprecate models X, Y, Z because they are duplicative/stale/obsolete

## 5. Automate validation between legacy and new systems

As we discussed above, validation of parity between legacy and new systems is key to getting business to adopt the new platform. It is also the most time-consuming part of the migration process.

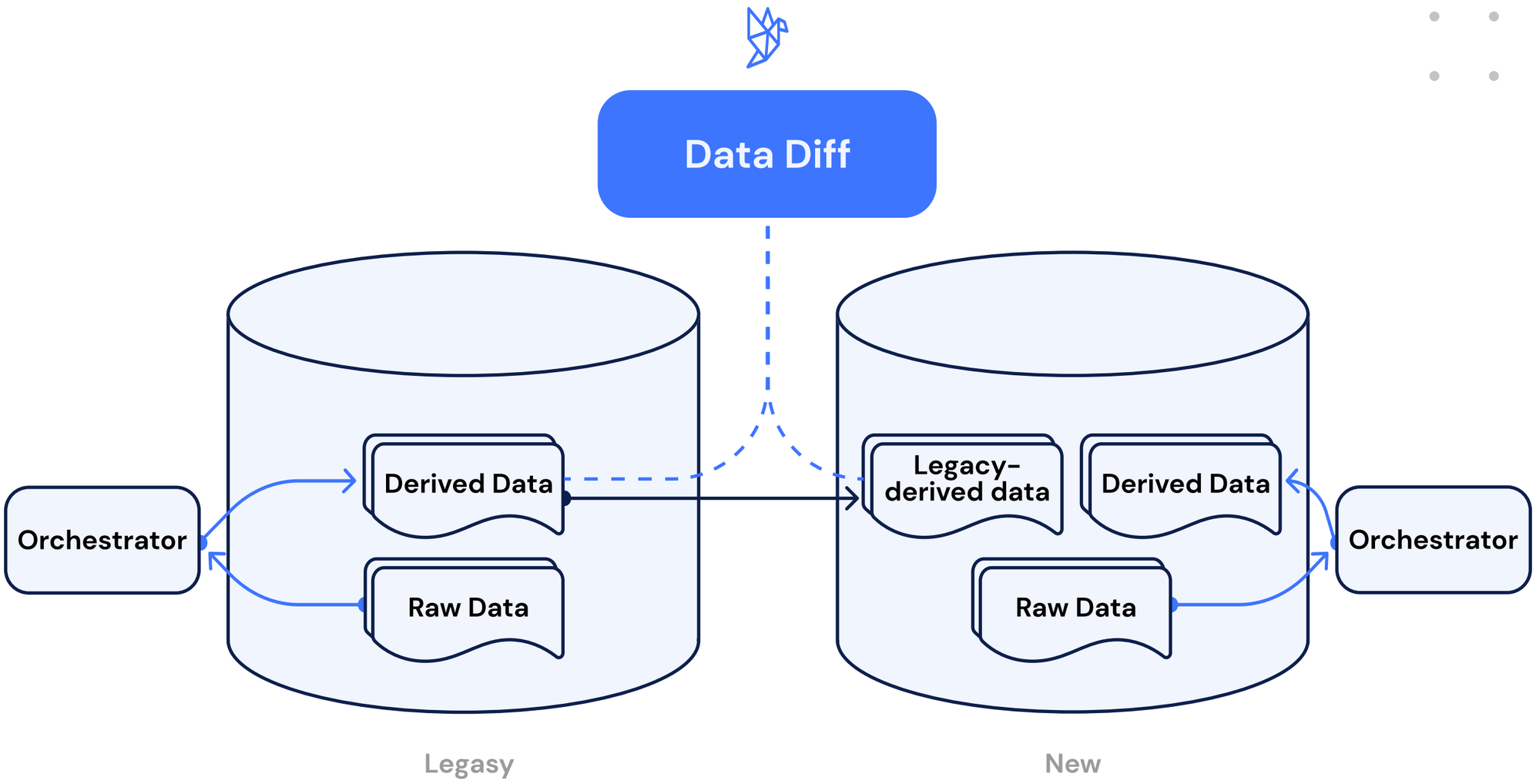
67% of respondents in our recent poll said the hardest, most tedious part of a migration was validating the legacy system's data with their new system's data. Which is where **data diffing** comes in.

| | order_value | | order_type | | total_value | |
|---|---|---|---|---|---|---|
| | 91.6% match | | 94% match | | 91.6% match | |
| | Oracle | Snowflake | Oracle | Snowflake | Oracle | Snowflake |
| 65248740 | 58.00 | 58.25 | Credit Card | Credit Card | 58.00 | 58.25 |
| 65248741 | 11.25 | 11.25 | Cash | Cash | 11.25 | 11.25 |
| 65248742 | 9.50 | 9.50 | Credit Card | Credit Card | 9.50 | 9.50 |
| 65248743 | 33.00 | 33.00 | Cash | Cash | 33.00 | 33.00 |
| 65248744 | 65.00 | 65.10 | Credit Card | Credit Card | 65.00 | 65.10 |
| 65248745 | 9.50 | 9.50 | NULL | Credit Card | 9.50 | 9.50 |
| 65248746 | 9.50 | 9.50 | Credit Card | Credit Card | 9.50 | 9.50 |
| 65248747 | 9.50 | 9.50 | NULL | Credit Card | 9.50 | 9.50 |
| 65248748 | 9.50 | 9.50 | Cash | Cash | 9.50 | 9.50 |

During the validation phase of a migration, the best (and fastest) way to ensure 1-1 table parity between your legacy and new databases is by performing a data diff: a row-by-row comparison of two tables that lets you know what's been changed, deleted, or added between the two. This enables data teams to identify critical changes to your data and guarantee data quality.

You can use Datafold Cloud's cross-database **data diffing** to detect and explain any inconsistencies that are present between the legacy and new version of your tables. Two of the most common issues related to database migrations are:

1. *Is the data the same across the two systems?*
2. *If inconsistencies exist, exactly which rows and values are different?*




By diffing your data, you can compare source and target data within and across databases, making it easy to answer these questions.

Diffing your data during a migration allows you to validate the parity between hundreds or thousands of tables within minutes—whether these tables are in the same data warehouse or in different ones.

By using Datafold Cloud's data diff during a migration, you know immediately if data you've copied, switched up the transformation workflow for, or received from a new source is different from the original data. Your team has **the agency** to quickly determine if these changes are expected and acceptable, or if something has gone wrong during the migration.

> "Seeing statistics and visualizations on how new model data compares to existing model data makes it easy to verify our migration SLAs with stakeholders," said Adam Underwood, Staff Analytics Engineer at Eventbrite. "This has saved incredible amounts of time and drive efficiency for us."

## 6. Leverage user acceptance tracker to signal project completion

How do you know when your database migration is complete? It's typically signaled by turning off the old system. Before you can do that, though, you must gain the trust and approval of your business stakeholders. That means they must first sign-off on the change and agree to adopt the new system before you can transition away from the old one.

We recommend running a user acceptance testing phase in which stakeholders provide their official sign-off, table by table, in a spreadsheet. See an example here.

**To see how Datafold Cloud can help you with real cost-savings during your migration, check out the Datafold Migrations ROI calculator.**

# How To Migrate A Database: A 4-Step Process

## 1. PLAN

Scope with column-level lineage, then prioritize and deprecate assets

## 2. MIGRATE

Lift and shift the data and code with a SQL dialect translator

## 3. VALIDATE

Verify the data and code matches the old source of truth with Datafold Cloud's data diffing abilities

## 4. REPEAT

Follow the three steps again for the next data asset type (tables, models, dashboards)

# 2 Real-Life Examples To Learn From

How Faire migrated from Redshift to Snowflake 6 months faster using Datafold's Data Diff feature

[Read the case study →]

***"No question, I would recommend Datafold for any large-scale migration."***

— Jon Medwig, Staff Data Engineer, Faire

How Eventbrite saved hours validating each model during their migration from Presto to Snowflake using Datafold

[Read the case study →]

***"Seeing statistics and visualizations on how new model data compares to existing model data makes it easy to verify our migration SLAs with stakeholders."***

— Adam Underwood, Staff Analytics Engineer, Eventbrite

# Conclusion

There’s no need to reinvent the wheel when it comes to a data migration. Data engineers at companies like Lyft, Faire, and Eventbrite have shared their hard-earned insights to help you navigate your database migration with confidence. By following the best practices outlined in this ebook, you can complete the project on time and budget—with maximum acceptance from your key stakeholders.

Of course, if you have any questions along the way, our team of database migration experts would be happy to jump on a call.

**Book a consultation**

# Key Takeaways

1. Evaluate technology vendors against five key criteria—scalability, user experience, interoperability, trajectory, and cost.

2. Identify your most and least important assets for migration using column-level lineage to help plan and prioritize.

3. Lift and shift the data to avoid considerable downtime and use a SQL translator to achieve accurate and efficient migration across systems.

4. Document your strategy and action plan up front to win buy-in from stakeholders and align the team.

5. Use data diffs to quickly validate data across systems seamlessly to massively accelerate your migration project.

6. Signal stakeholder approval and project completion through user acceptance testing.

# Data Migration Best Practices: A How-To Guide for Data Teams

*The technological and organizational best practices for faster (and less awful!) migrations.*

Datafold, 2023

Datafold