Bigeye

The Ultimate Guide to Data Observability

Demystifying Data Observability for the Enterprise

Table of contents

3 Introduction

4 What is Data Observability?

5 Why is Data Observability important?

6 What are the benefits of Data Observability?

7 What are the benefits of Data Observability?

8 What's the difference between Data Observability and...

10 How does Data Observability work?

11 Basic Data Observability concepts

17 Architecture and location within the Data Stack

18 Use cases to solve with Data Observability

19 Common signs that it's the right time for Data Observability

20 You've moved to a hub-and-spoke Data Team Structure

21 Signs you should reevaluate your current solution

22 How to evaluate Data Observability Vendors

23 Best practices for Data Observability

25 Getting started with Data Observability

Introduction

Welcome to Your Guide on Data Observability.

In today's data-driven world, organizations are increasingly reliant on vast amounts of data to drive strategic decision-making, enhance user experiences, and streamline operations. However, with this dependence comes the critical need to ensure that the data being utilized is accurate, reliable, and timely. This is where data observability comes into play.

Data observability is the practice of understanding and monitoring the health and state of data across its entire lifecycle. From ingestion to processing and final utilization, data observability provides a comprehensive view into how data is moving, where it is going, and whether it meets the quality standards required for effective use. This ebook aims to demystify the concept of data observability, highlight its importance, and offer practical insights into its implementation.

This ebook is designed to provide you with a comprehensive understanding of data observability, including its importance, benefits, and best practices. Whether you're new to the concept or looking to deepen your knowledge, this guide will equip you with the information you need to effectively implement data observability within your organization.

Data Observability is the practice of understanding and monitoring the health and state of data across its entire lifecycle. From ingestion to processing and final utilization, data observability provides a comprehensive view into how data is moving, where it is going, and whether it meets the quality standards required for effective use.

What is Data Observability?

Definition

Data observability refers to the ability of an organization to see and understand the state of their data at all times. By "state" we mean things like: where is it coming from and going within our pipelines, is it moving on time and with the volume we expect, is the quality high enough for our use cases, and is it behaving normally or did it change recently?

Observability platforms aim to give a continuous and comprehensive view into the state of data moving through data pipelines, so questions like these can be easily answered.

Once data teams unlock these activities, they can systematically understand when, where, and why data quality problems occur in their pipelines. They can then prevent those problems from impacting the business, and work to prevent them occurring in the future!

Data observability unlocks these basic activities, so it's the first stepping stone toward every organization's ultimate data wishlist: healthier pipelines, data teams with more free time, more accurate information, and happier customers.

Here are some questions you could answer with Data Observability:

Is the customer's table getting fresh data on time or is it delayed?

Do we have any duplicated shopping cart transactions and how many?

Was the huge decrease in average purchase size just a data problem or a real thing?

Will I be impacting anyone if I delete this table from our data warehouse?

Common Data Observability activities include:

Monitoring the operational health of the data to ensure it's fresh and complete, detecting and surfacing anomalies that could indicate data accuracy issues, mapping data lineage to upstream tables to quickly identify the root causes of problems, and mapping lineage downstream to analytics and machine learning applications to understand the impacts of problems.

Why is Data Observability important?

Organizations push relentlessly to better use their data for strategic decision making, user experience, and efficient operations.
All of those use cases assume that the data they run on is reliable.

The reality is that all data pipelines will experience failures. It's not a question of if, but when, and how often. What the data team can control is how often issues tend to occur, how big the impact, and how stressed out they are when resolving these failures.

A data team that lacks this control will lose the trust of their organization, therefore limiting organizational willingness to invest in things like analytics, machine learning, and automation. On the other hand, a data team who consistently delivers reliable data can win the trust of their organization, and fully leverage data to drive the business forward.

Data observability is important because it is the first step toward having the level of control needed to ensure reliable data pipelines that win the trust of the organization and ultimately unlock more value from the data.

Who uses Data Observability?

Data observability can touch many departments within an organization. But if you're wondering who it's for and who works with it, let's explore some context. Following, we'll explore the relationship between data observability and some common data roles within teams.

What are the benefits of Data Observability?

What do you get once you have total observability over your data pipelines? The bottom line is that the data team can ensure that data reaching the business is fresh, high quality, and reliable—which unlocks trust in the data.

Let’s break down the tangible benefits of data observability a little further:

01 Decreased impacts from data issues—when problems do occur, they’ll be understood and resolved faster; ideally before they reach a single stakeholder. Data outages will always be a risk, but with observability, their impacts are greatly reduced.

02 Less firefighting for the data team—you’ll spend less time firefighting data outages, and being reactive. That means more time building things, creating automation, and the other fun parts of data engineering and data science.

03 Increased trust in the data by stakeholders—once they stop seeing questionable data in their analytics, and stop hearing about ML model issues, they’ll start trusting the data and assuming it’s good for making decisions with or integrating into their products and services.

04 Increased investment in data from the business—once stakeholders can trust the data, they can feel comfortable using data in more places across the business, which means allowing a bigger budget on data and the data team.

The history of Data Observability

The concept of “data observability” began emerging in the late 2010s, driven by the need for more robust data quality management within large-scale data environments. Companies like Uber, Netflix, Airbnb, and Lyft pioneered internal solutions aimed at improving data pipeline health and reliability. Initially, these teams developed pipeline testing tools—basic systems that allowed engineers to manually check whether their data met expected conditions.

As data ecosystems grew in complexity, these testing tools quickly revealed their limitations. Rule-based systems like Informatica or dbt tests required manual configuration, making it difficult to scale and adapt as data environments evolved. Maintaining these rules became time-consuming, and legacy systems struggled to handle modern cloud data environments, which operate at massive scales and across diverse pipelines. The need for automation, flexibility, and deeper insights led to the development of true data observability platforms.

Unlike traditional data quality products from that era, modern observability platforms like Bigeye were built specifically for cloud data environments. They automate the detection and diagnosis of issues, providing continuous monitoring and instant coverage for entire data warehouses. This shift from manual testing to observability is crucial: without observability, teams are often blind to critical issues that can compromise AI models, regulatory compliance, and business decisions.

While many companies now embrace tools like data warehouses, orchestration platforms like Airflow, and transformation tools like dbt, data observability remains wrongly viewed by some as a “nice-to-have.” In reality, it’s the key to ensuring your data is actually fit for use. By continuously monitoring data in use, observability offers the visibility needed to detect anomalies, trace issues back to their source, and maintain reliable, trustworthy data pipelines.

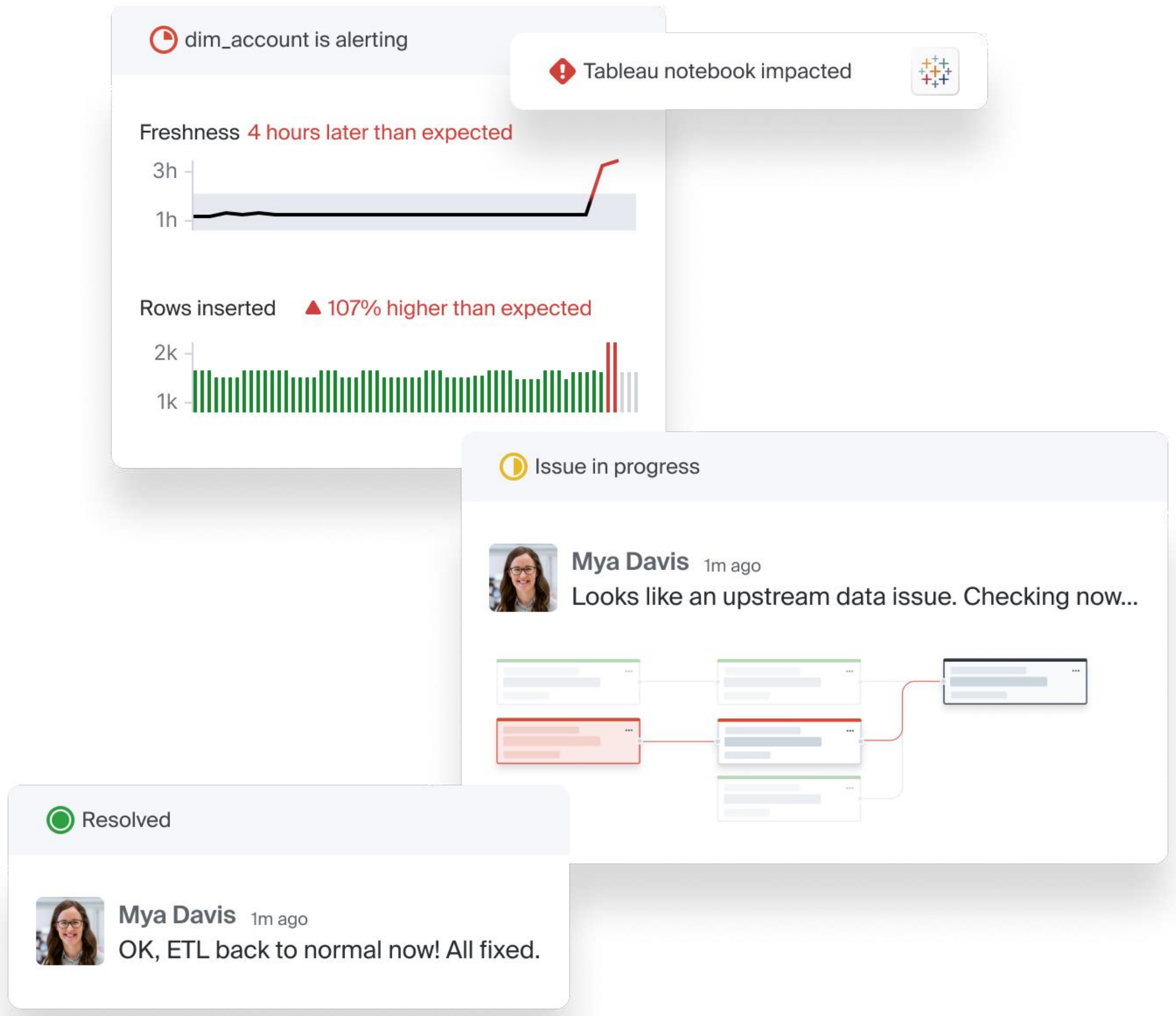
What's the difference between Data Observability and...

Data Testing?

Testing is a manual check for specific conditions, like whether a data table has non-zero entries or if a pipeline produces the expected output. While testing is crucial for catching known issues, it's limited in scope and manual in nature. In contrast, data observability is broader and automated, providing continuous monitoring and analysis of your data pipeline. Testing feeds into observability, but they work best together for maintaining data health.

Data Monitoring?

Data monitoring tracks predefined metrics over time, like a dashboard displaying system stats. It's useful for keeping an eye on known conditions but lacks depth. Data observability goes beyond this by diagnosing root causes and offering insights on unexpected patterns. Think of monitoring as checking your car's dashboard, while observability is like running a full diagnostics test.


dim_account is alerting
Tableau notebook impacted
Freshness 4 hours later than expected
3h
1h
Rows inserted
107% higher than expected
2k
1k
Issue in progress
Resolved

Mya Davis 1m ago
Looks like an upstream data issue. Checking now...

Mya Davis 1m ago
OK, ETL back to normal now! All fixed.

Data Reliability?

Data reliability is the goal, while data observability is the approach that helps you achieve it. By implementing observability practices, you ensure your data systems are consistently reliable.

Machine Learning Observability?

While data observability tracks data health metrics like freshness and accuracy, machine learning observability focuses on monitoring model performance metrics like drift and anomalous behaviors. Both are essential, but they target different layers of your data ecosystem.

Infrastructure Observability?

IT teams use infrastructure observability tools to track server and application metrics like uptime and error rates. Data observability is inspired by this but deals with more dynamic metrics that change significantly over time, making it unique to data teams.

Data Quality?

Data quality refers to the overall health of your data—how clean and reliable it is. Data observability, however, continuously surveys your data pipeline to detect and proactively address issues, ultimately supporting better data quality.

For more definitions,
check out our complete
Data Observability dictionary.

How does Data Observability work?

In general, a good data observability platform should monitor overall warehouse health (freshness, volume, accurate formatting), data quality issues in the tables themselves (anomalies, outliers, potentially erroneous entries), and where problems and improvements will have an impact on the operation (lineage, root cause analysis).

There are two types of Data Observability:

01 Warehouse metadata monitoring

02 Deep statistical monitoring

General warehouse health visibility is achieved through warehouse metadata monitoring. It's fairly straightforward and low impact to your data environment, so it's easy to apply across the entire warehouse.

Deep statistical monitoring goes deeper. It is achieved by connecting to a read-only account and applying metrics to your data. This approach is very similar to connecting a BI tool. But a good data observability tool won't actually copy any of your data. It will simply monitor the source and store observability information—limiting operational strain on the database and minimizing security risks.

Companies generally only apply deep data quality checks on the most critical tables. For most companies, that's about 20% or less of all the data in their warehouse!

Common deep monitoring attributes to track include completeness, duplication, format errors, outliers, and distributional statistics. These attributes can't be collected from metadata. They require direct interrogation of the data itself. This is necessarily more impactful on the warehouse, but also enables the detection of issues inside the datasets themselves.

Finally, a data observability tool will parse logs and use AI to map out the data model and flow of your environment. This process helps you understand the potential impacts of changes upstream and down.

NOTE:
In both cases, only aggregates need to be returned and stored in the data observability platform. Raw data isn't needed at this stage, and doesn't need to leave the data source.

Basic Data Observability concepts

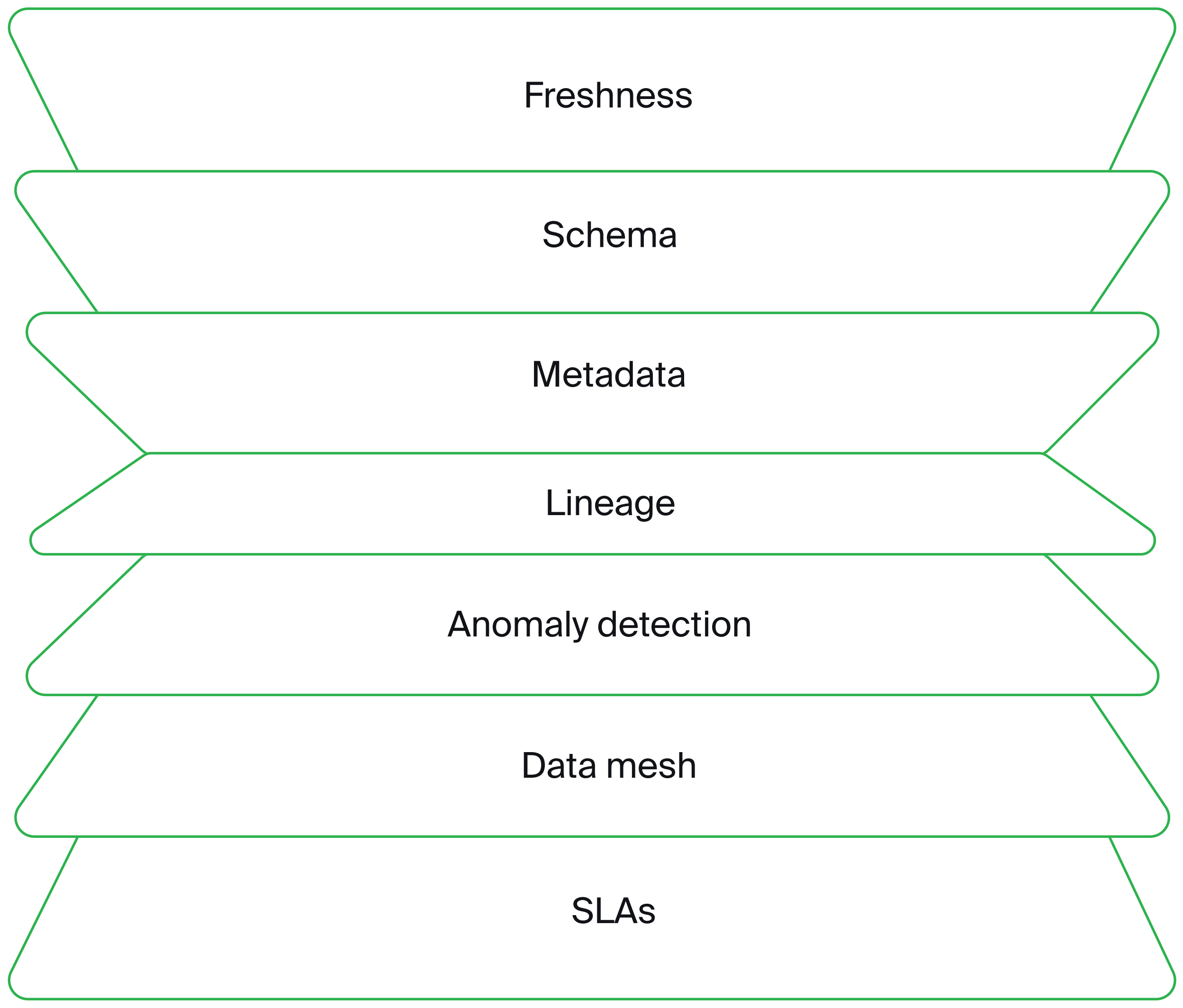
If you're completely new to data observability, there are some foundational concepts that you can digest. The following summaries are meant as bite-sized primers to help you familiarize yourself with the core aspects of a data observability system.

1. Freshness

Data freshness refers to how up-to-date data is, e.g. the amount of time since a data table was last refreshed. Your data's freshness is one of the main things that an observability platform will track; ensuring that no data goes stale and out-of-date.

2. Schema

A database schema is an abstract design that represents the storage of your data in a database. It describes both the organization of data, and the relationships between tables in a given database.



3. Metadata

Metadata is data about data. In the data observability context, metadata tells you about what's being done to the data by the infrastructure, e.g. an INSERT to append new rows to a table, or an Airflow job that failed and didn't write anything to its intended destination.

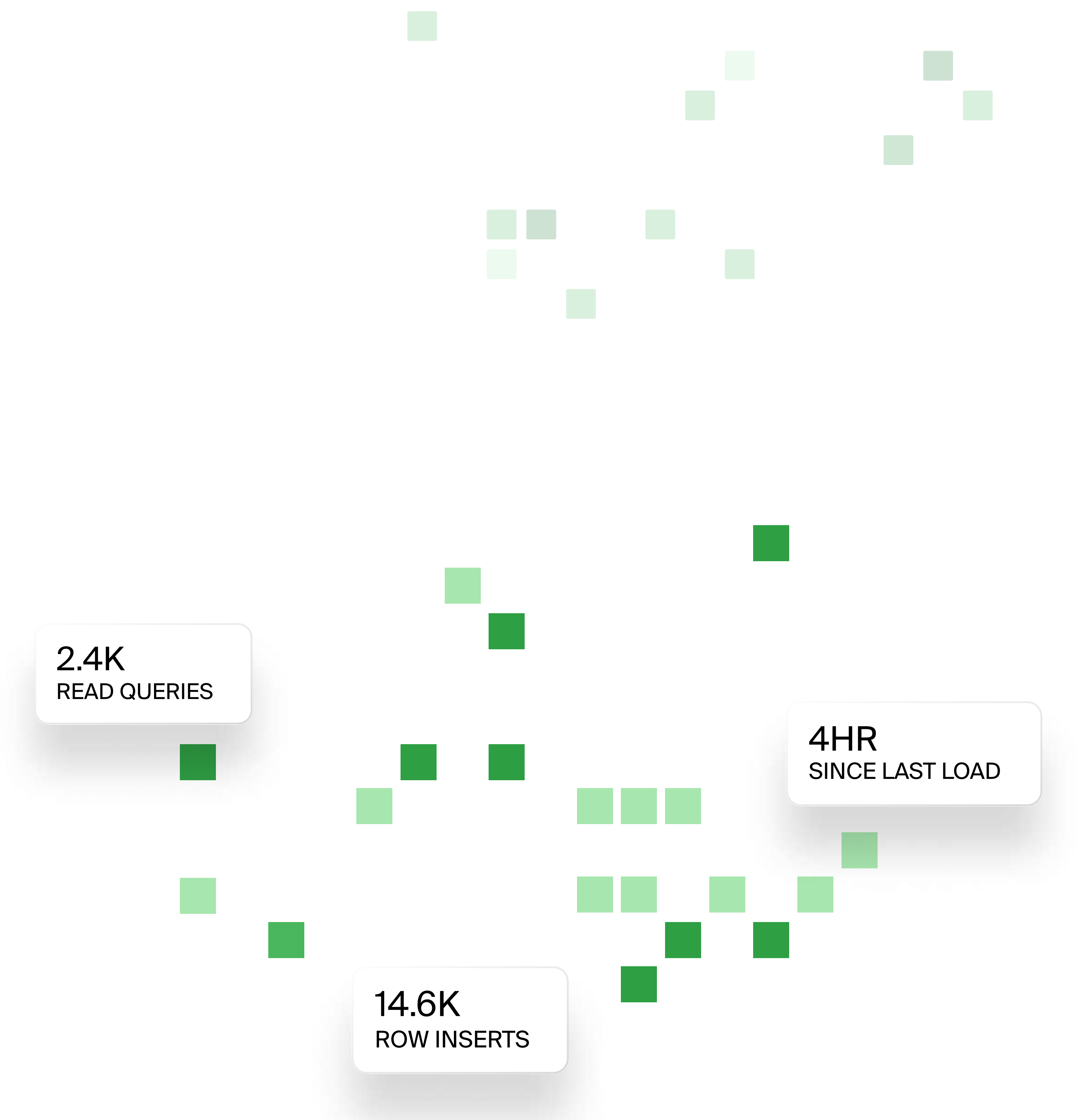
Metadata can give you key information about, for example:

01 **Data freshness**
e.g. time since a data table was last refreshed

02 **Data volume**
e.g. number of rows inserted per day

03 **Read query volume**
e.g. number of queries run per day

Metadata is important for observing the state of your data system. Out-of-date data (for instance in dashboards) is often extremely visible to non-technical leadership, and impedes their decision-making ability. Meanwhile, a massive drop in data volume or data query volume almost certainly indicates that an upstream or downstream dependency is down.



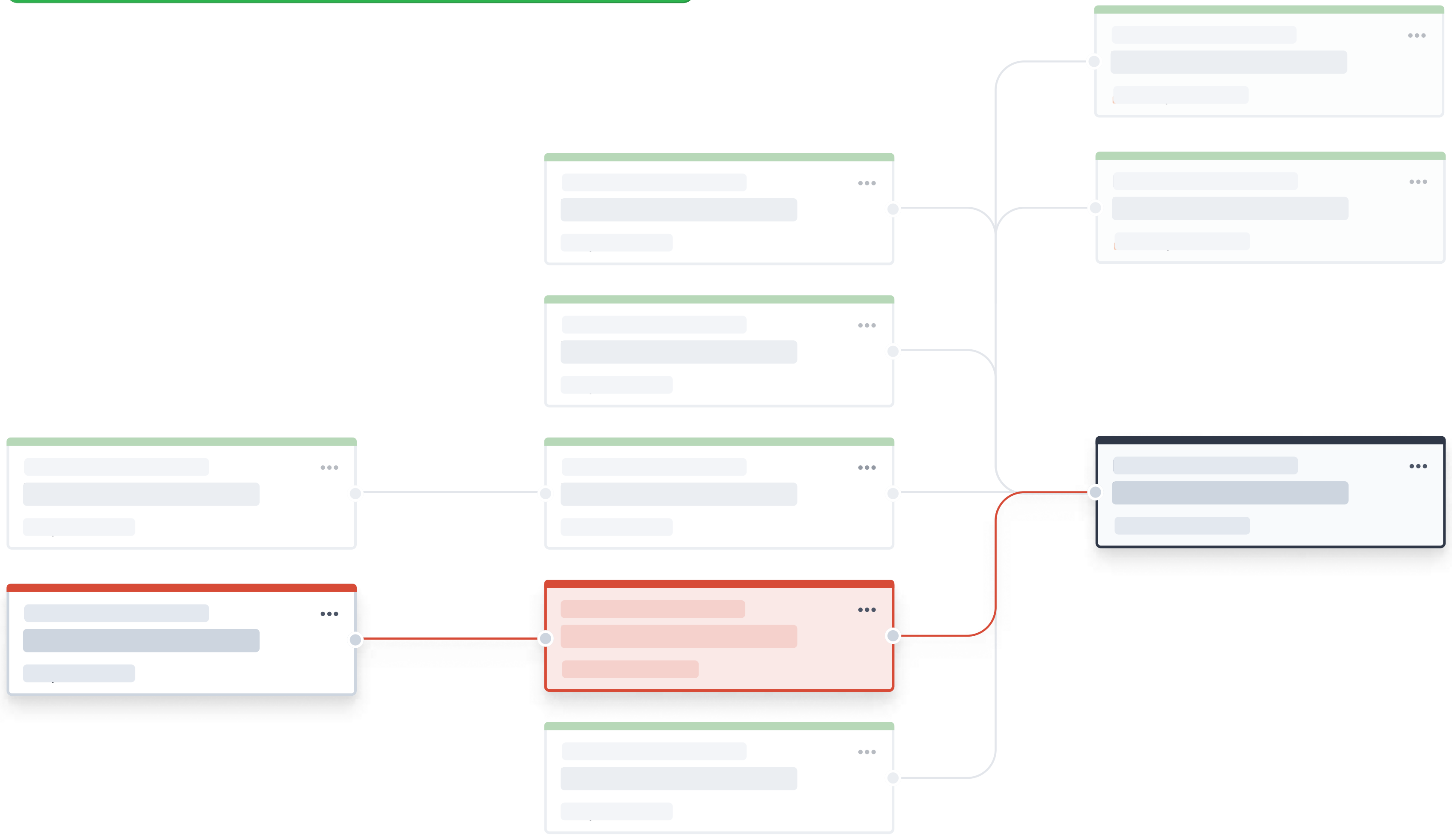
4. Lineage

A data breakage or outage happens somewhere along the pipeline. The first question is, where? Data lineage helps you find out, by feeding information about which upstream and downstream sources were impacted. It also ties that information back to the teams that are generating and accessing the data. Lineage is a key piece of any data observability program. When a data issue occurs, your lineage serves as a map for how governance, business, and technical teams are impacted.

To be more specific, data lineage is the path that data takes through your data system, from creation, through any databases and transformation jobs, all the way down to final destinations like analytics dashboards and feature stores. Data lineage is an important tool for data observability because it provides context - it tells you:

Whether problems are localized to just one dataset, or cascade down your pipeline

For each data pipeline job, which dataset it's reading from and the dataset to which it is writing



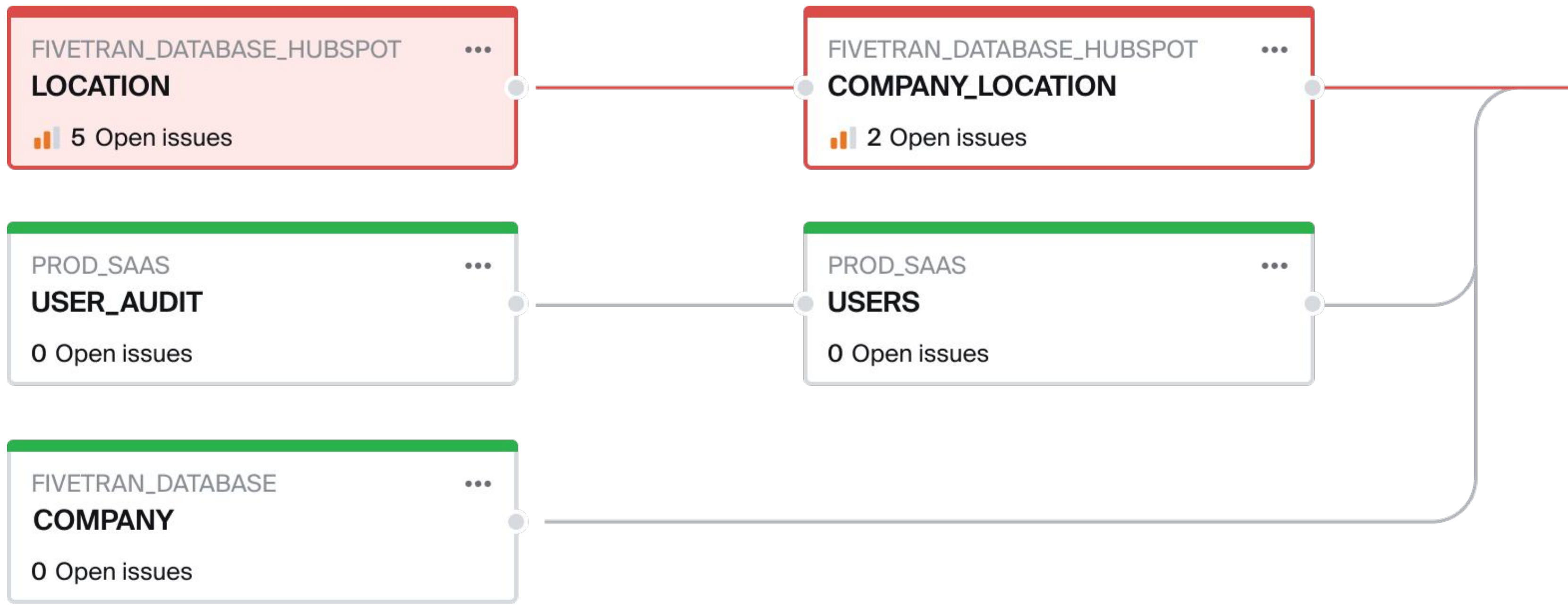
This helps to answer questions about where data problems originate, and how widespread their impact is:

If I change the schema of this table, what other tables will start having problems?

If I see a problem in this table, how can I identify where that problem originated, and how far it propagates?

Lineage is most often collected by parsing the logs of queries that write into each table. By inspecting what's happening inside the query, you can see which tables are being read from, and which tables are being written into.

But lineage can (and should) go further than just table-to-table or column-to-column relationships within a single source. Companies like AirBnB and Uber have been modeling lineage all the way upstream to the source database or Kafka topic, and all the way downstream to the user level, so they can communicate data problems or changes all the way up to the relevant humans.



5. Anomaly detection

Detecting data points, events, and/or information that falls outside of a dataset's normal behavior. Anomaly detection helps companies flag areas that might have issues in their data pipelines.

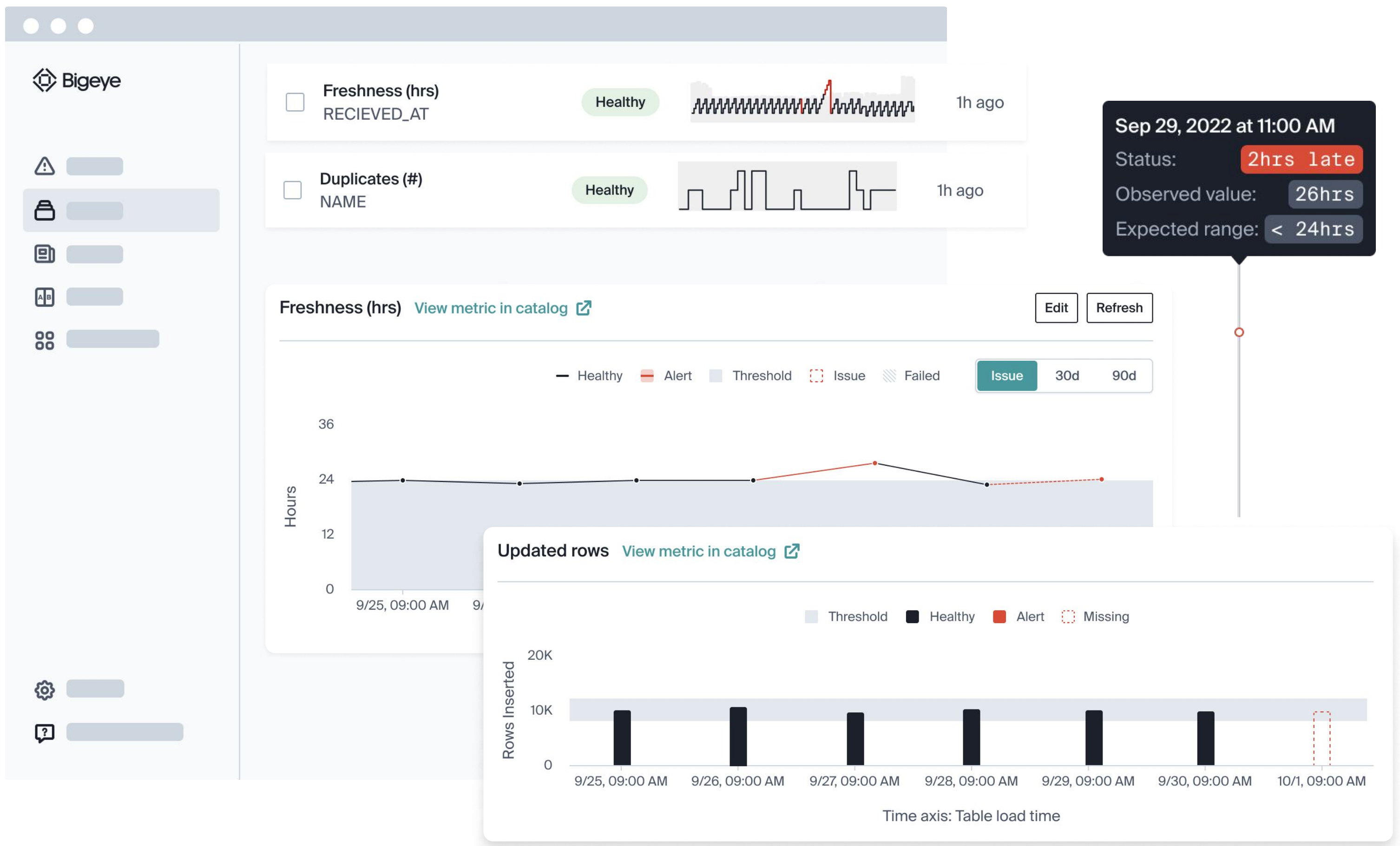
The anomaly detection system will need to learn the historical patterns present in each data quality attribute, learn what abnormal behavior looks like, and ultimately fire alerts that indicate real issues while ignoring behavior that's slightly off but not indicative of a real problem. This can be especially difficult when there are hundreds or thousands of data quality attributes being tracked simultaneously.

Naive techniques like gaussian models—that simply look at a number of standard deviations above or below the historical mean—fall apart in many commonly occurring time series patterns. A good model will need to adapt to various patterns that can regularly occur in these metadata attributes over time.

For a **deeper dive into anomaly detection, read here.**

6. Data mesh

A data mesh is a type of data architecture. It assigns ownership to various parts of data pipeline based on specific business teams - Marketing, Sales, Product, etc. A data mesh eliminates bottlenecks associated with one monolithic data system, and makes data observability more effective by pinpointing data changes, movement, access requests, and other events at a more granular level within the data pipeline.



7. SLAs

Service Level Agreements are a foundational part of data observability. SLAs were originally designed to clarify and document expectations between a service provider and its users. In a data observability context, they allow for clear communication about what "good" and "not good" data looks like!

They're built up in three stages:

SLIs: Service Level Indicators—what attributes will be tracked and what are acceptable levels

SLAs: Service Level Agreements—who does what if the SLOs are not met

SLOs: Service Level Objectives—how often will the SLIs be within their expected range

For example:

- **SLI:** Percent of null user_uuids per partition <= 0.5% in last 2 partitions
- **SLO:** **99%** as tracked daily over a trailing 30 day window
- **SLA:** Data engineering will halt pipeline changes until resolved and SLO back within 99%

The development of the SLIs, SLOs, and SLA creates a clear framework for data consumers and data teams to align on exactly what "high quality" data means, and what will be done if that definition isn't being met. It prevents ambiguity and eliminates arguments during high pressure situations when something does happen to go wrong.

For a **deeper dive into SLAs, read here.**

Architecture and location within the Data Stack

Data observability systems usually consist of the following elements:

The control plane
The interface where everything is managed from.

The collection system
How the metrics and metadata get collected from data sources.

The anomaly detection system
Takes in metrics, builds models, and fires alerts.

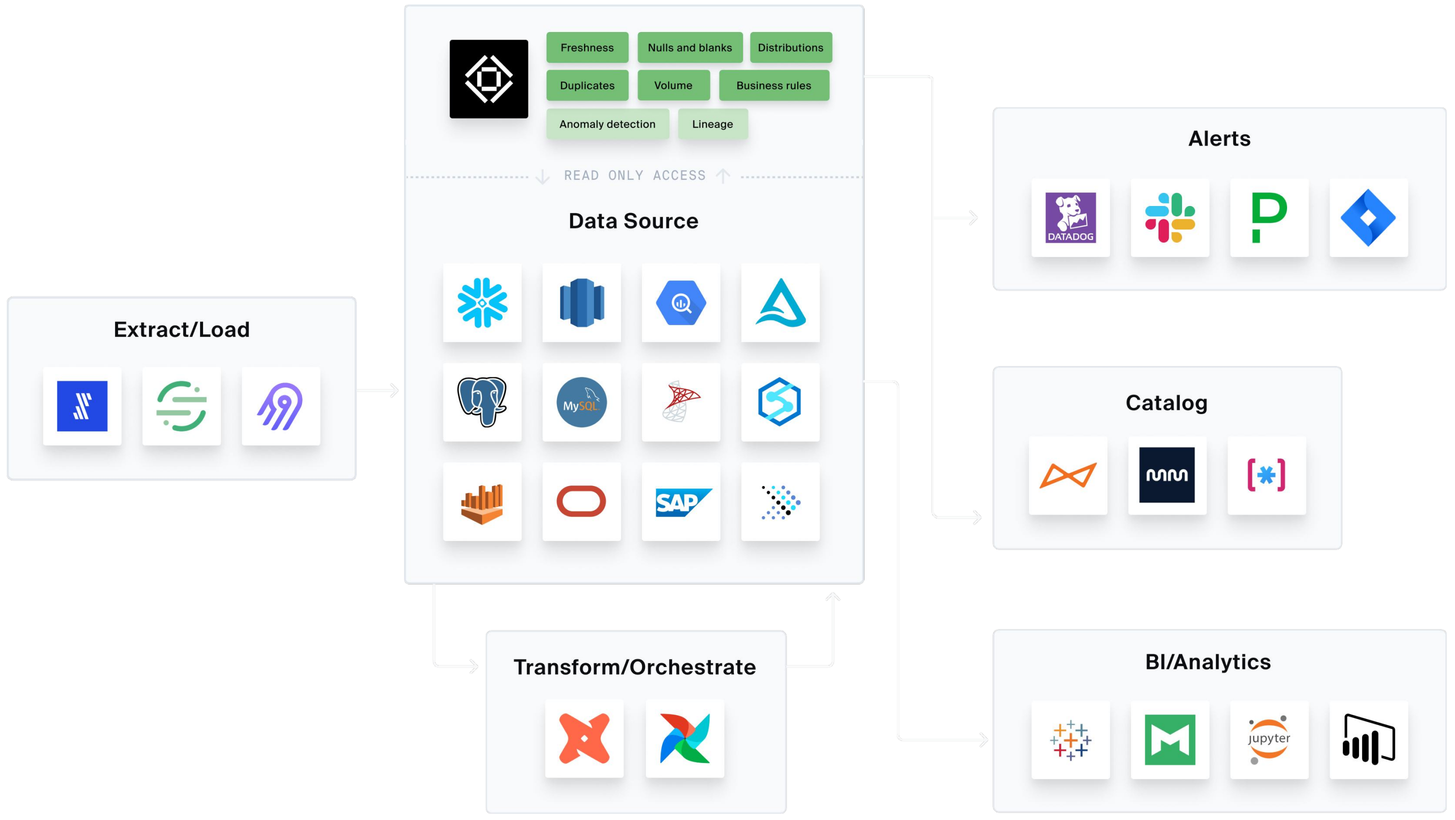
The notification connectors
Outputs to Slack, PagerDuty, data catalogs, etc.

Queries are run on the data source, the metrics or metadata are summarized, and sent back to the control plane where they're stored. This means raw data stays on the data source, and only aggregates are transmitted back to the control plane.

Once stored in the control plane, the metrics and metadata results are fed to the anomaly detection system so anomaly detection models can be trained on them and alerts can be fired if needed.

Finally, if an alert is needed, the control plane needs to be connected to the desired alert destination, like a Slack workspace, a PagerDuty workspace, a data catalog like Alation, or an analytics tool like Tableau.

For a deeper understanding of where data observability fits within your data stack—whether you're using a fully modern setup or working with legacy systems—check out these helpful resources. Our blog post on Data Observability in the Modern Data Stack breaks down how observability integrates with today's cloud-native tools. If you're navigating a more complex environment, you'll want to explore Navigating a Non-Modern Data Stack to see how observability can still play a critical role in maintaining data reliability.



Use cases to solve with Data Observability

When should you use data observability?
When do teams start to explore data observability platforms, and are there any universal indicators that data observability should deploy? Let's walk through the various use cases for which we prescribe data observability as the cure.

01

Keeping analytics accurate and trustworthy
Executives at large organizations regularly complain that "they don't trust the data". This is often because the data is internally inconsistent, or because there are discrepancies between the expected numbers and what the dashboard is showing. Ensuring that analytics dashboards and reports display accurate data is probably the top use case for data observability.

02

Protecting machine learning performance
Machine learning models are "garbage in, garbage out." In order for recommendation systems, fraud detection systems and computer vision systems, to generate accurate recommendations and predictions, the data inputs must be accurate and up-to-date. Data observability helps ensure these standards.

03

Accelerating ETL/ELT development velocity
Data model blue-green testing is a framework for deploying changes to data pipeline jobs. The framework expects two schemas in your database, one staging (blue), and one production (green). When you make a change to your job in your data pipeline, you initially run the changed job only in the staging schema. You then compare the outputted data to what is in the production schema. The only differences should be the expected ones from your change.

04

Accelerating data warehouse migrations
When you invest in a cloud data warehouse, you want to get up and running and start seeing value quickly. Unfortunately, migration complexity and manual validation processes can often kill momentum and force data teams to burn time and resources chasing migration issues instead of adding value to the business. Data observability can reduce the friction and produce faster, smoother migrations by replacing manual checks with automated migration validation and comprehensive reporting—helping ensure every row of data is delivered safe and sound.

Common signs that it’s the right time for Data Observability

You just experienced a high-severity Data Outage

Data quality is a traditional, all-encompassing term that is focused on fixing data issues in a reactive manner. Data quality refers to the general state of your data: how healthy is it?

In contrast to data quality, data observability constantly surveys the state of the data pipeline and proactively diagnoses issues. Data observability platforms can help to ensure data quality.

Your pipelines have gotten complex

A lot of your teams are all about data. Where does it come from, where is it going, what is it telling you about how to run your business better, and reach more customers?
Data environments are constantly growing, shifting, and evolving. They truly take on a life of their own.

The “modern data stack” involves more tools now than ever before. That means more pipelines, more tables, and more opportunities for failure or disruption along with them. At all stages of the data pipeline, there are complex transactions occurring. Data is flowing between storage, processing, transformation, modeling, and business intelligence.

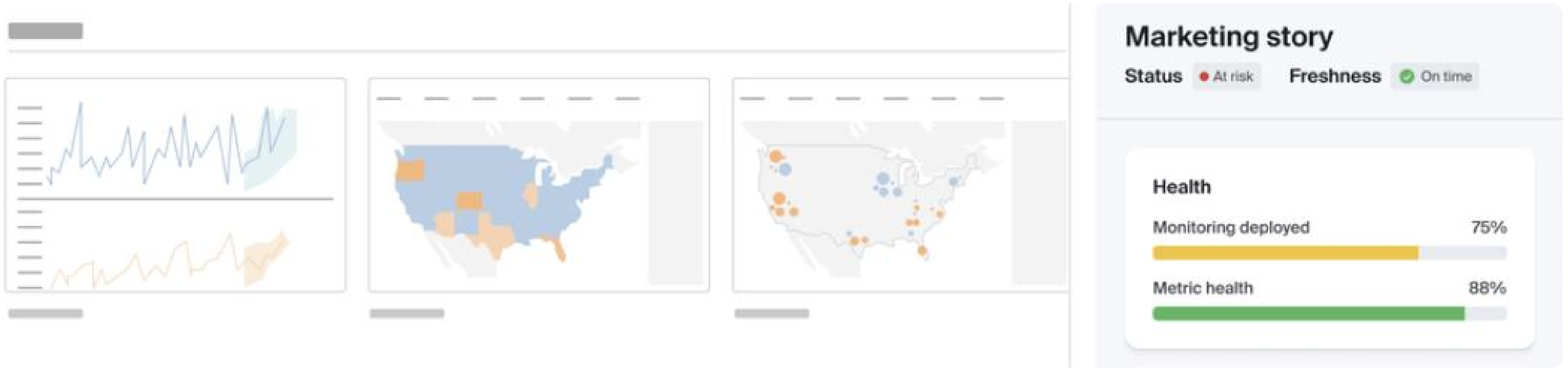
Teams can’t wait to be blindsided by inaccurate, broken, or stale data.
One schema change can cause a furious uproar and catastrophic consequences. Change means growth, but it also means unpredictability. Data observability is technology’s answer to that unpredictability; data observability platforms introduce predictability and reliability back into your complex data pipelines. You can’t manually keep data catalogs up-to-date with spreadsheets and the occasional debriefing meeting. You need sharper visibility into your data pipelines and anomalies as soon as they occur.

You’ve moved to a hub-and-spoke Data Team Structure

Data touches a lot of hands on its journey through your organization. As teams add data scientists, analysts, data reliability engineers, and business analysts, the ownership of data functions might shift or change hands, even within specific teams. And what about multiple teams that partially own some of the same data?

Data observability can help teams understand how work fits into the larger puzzle of data in your organization. Schema changes, new data sources, and pipeline additions are tracked and communicated with data observability. That way, teams can understand the impact of changes that feel minor, but might cause major ripple effects. Data observability is an effective communication tool; as your data writes a story, data observability serves as the transcript.

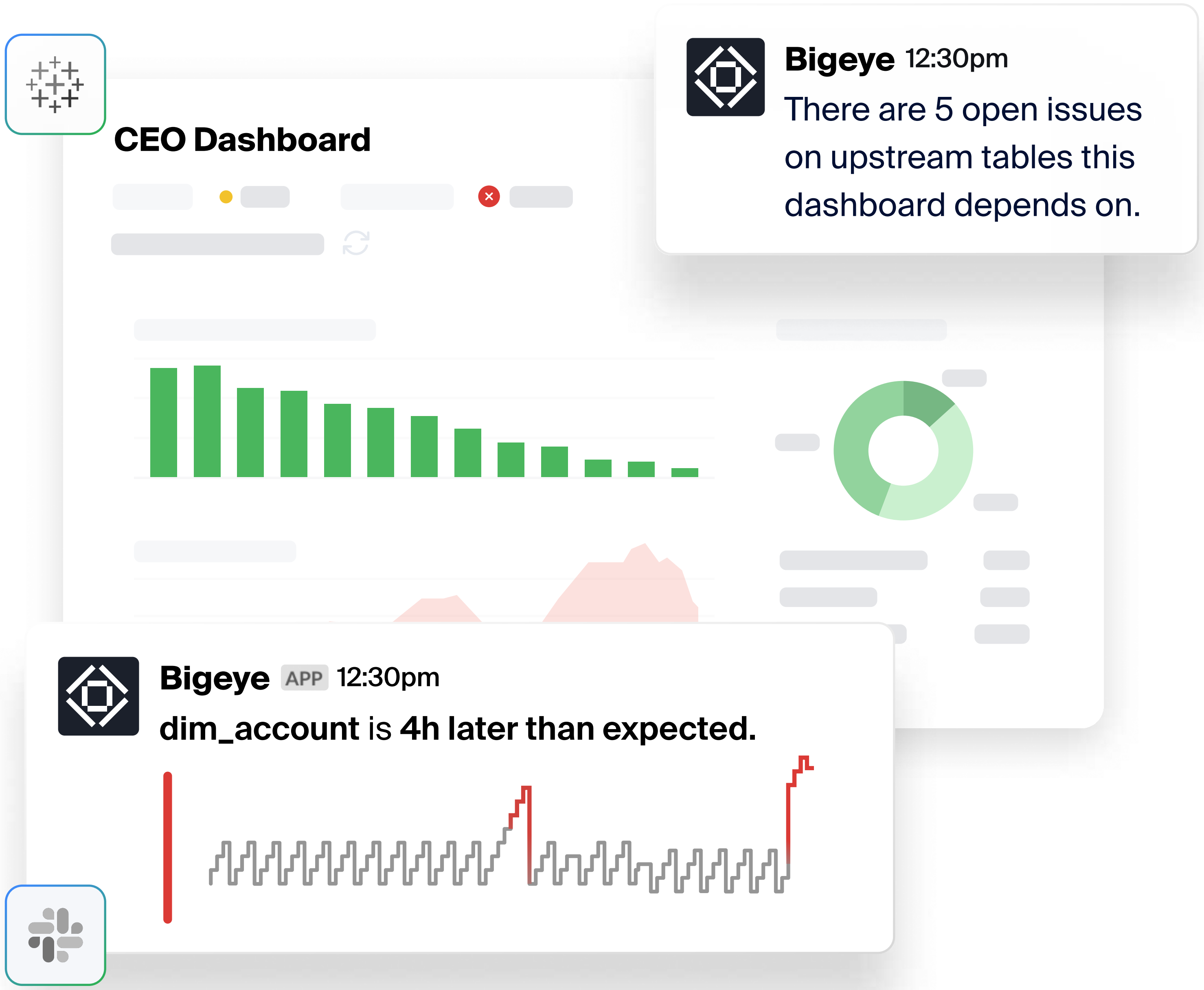
Data observability is an effective communication tool; as your data writes a story, data observability serves as the transcript.



Signs you should reevaluate your current solution

Your data is likely changing frequently: so should your data observability. On a periodic basis (maybe once a year), take stock of your data observability stack and tooling and re-evaluate whether it still suits your organizational needs.

This might mean conducting an audit of all the tables in your data warehouse, figuring out which of them might be decommissioned or modified, and removing or editing the associated metrics and alerts.



Bigeye 12:30pm
There are 5 open issues on upstream tables this dashboard depends on.

Bigeye APP 12:30pm
dim_account is **4h later than expected.**

How to evaluate Data Observability Vendors

Evaluating data observability platforms can feel overwhelming with so many features and vendors to choose from. Start by building a well-rounded evaluation team with representatives from your data engineering, analytics, and machine learning teams. Make sure you have a project leader who's both technical and business-savvy to steer the process, and don't forget to involve an executive sponsor early on to align with strategic goals and secure budget approval when it's time to move forward.

Once your team is set, focus on gathering the right requirements. This is where you really define what matters most to your organization. Whether it's deployment flexibility, advanced anomaly detection, or seamless integrations, tie each requirement back to a specific business challenge. Look closely at the solution's monitoring capabilities, how it handles alerting (too much noise is a dealbreaker), and how well it integrates into your existing data stack. The right vendor should offer a balance of out-of-the-box functionality and customizable features that fit your team's unique needs.

Lastly, keep scalability in mind. As your data environment grows, your observability solution needs to grow with it. Make sure the vendors on your shortlist can handle both current and future demands—whether that's monitoring hundreds of tables or supporting complex business logic rules. For a deeper dive into running a smooth evaluation, check out our original blog post and be sure to grab our downloadable Data Observability RFI to guide your vendor comparisons

Ready to Evaluate Vendors?

Read the Blog Post
Download the RFI Guide

Best practices for Data Observability

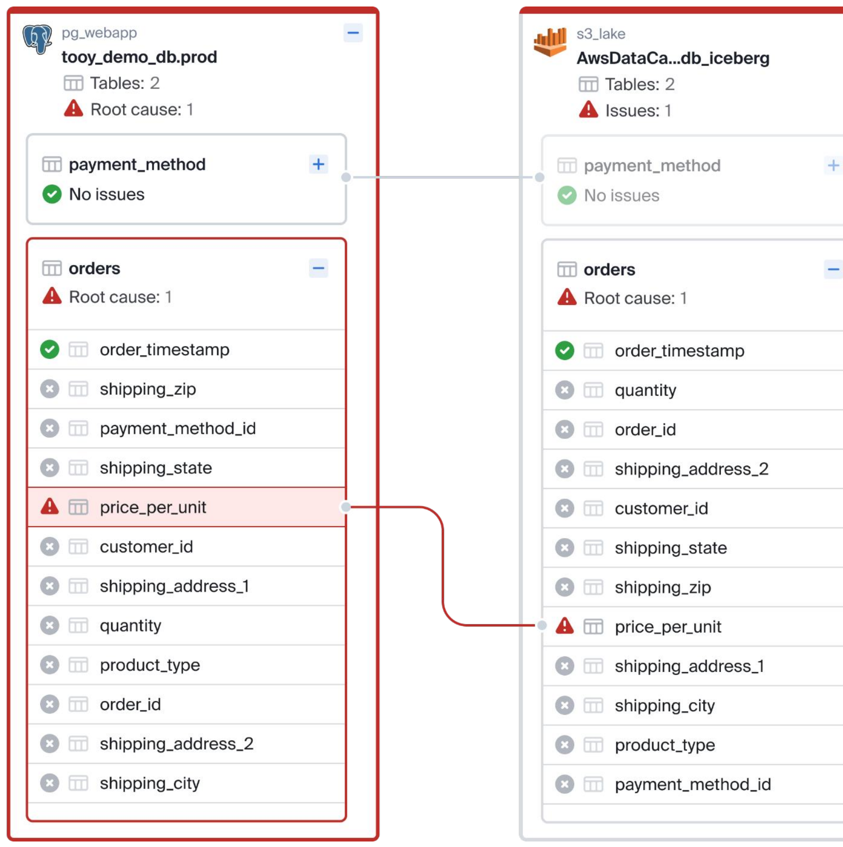
Whether you have a data observability solution in place or not, there are steps you can take to make your organization more observability-friendly. These best practices can be applied at any point in your data observability journey.

1. **Start defining the business-critical data**

   Before jumping into data observability, it's important to first go through your data and understand which data matters most, and which data can be ignored. After all, if you monitored all of your data at all times, you'd quickly reach your maximum capacity to deal with the onslaught of that information.

   Instead, look at what data is being consumed in the most business-critical ways. For example, the "Orders" table consumed by an analytics dashboard that is being regularly reviewed by executives is a high priority.

   Data lineage products that help visualize the path data takes through your data system, from creation, through any databases and transformation jobs, all the way down to final destinations like analytics dashboards and feature stores, can also be helpful.



Bigeye

2. Rollout data observability in a T-shape

Once you understand which data matters most, you can apply T-shaped monitoring. Everything in your organization doesn't need the same level of monitoring. Some areas of your data pipeline need special attention and care, and should be prioritized.

T-shaped Monitoring is an approach to data observability that tracks fundamentals across all your data, while applying deeper monitoring on the most critical datasets such as those used for financial planning, machine learning models, and executive-level dashboards.

Here's how it works:

01 Track data freshness first.

02 Then, select some business-critical datasets on which you can apply deep monitoring. Feel free to use a blend of metrics that Bigeye suggests for each table from its library of 70+ pre-built data quality metrics.

03 Add custom metrics with templates and virtual tables to ensure custom business logic is being monitored for defects.

For more information about T-shaped monitoring read on here.

3. Assign ownership over the stewardship the data pipeline

Each step in each data pipeline should be assigned to an owner. For example, if data moves from an online service to Kafka to Snowflake, undergoes transformations and lands in tables that are input data for fraud detection algorithm, the online-to-Kafka segment might be "owned" by the Kafka team, while the transformations for the fraud tables might be owned by the fraud team. The owner assigned is responsible for the data pertaining to that segment.

Getting started with Data Observability

Instructions

Use this worksheet to assess your current data observability needs and plan your implementation strategy. Answer the following questions to identify potential issues, understand their impacts, and develop a plan for continuous improvement.

This worksheet is designed to guide your team through the initial stages of implementing data observability and help establish a robust process for ongoing monitoring and improvement. Regularly revisiting and updating this worksheet will ensure your data observability practices remain effective and aligned with organizational needs

1. Identifying Potential Breaks

What can break?

(List the components, systems, or processes in your data pipeline that have the potential to fail)

2. Assessing Impact

What is impacted?

(Identify the downstream effects of each potential break, including affected departments, systems, and users)

3. Locating Breaks

Where do breaks occur?

(Pinpoint the exact locations in the data pipeline where breaks are most likely to happen)

4. Understanding Causes

Why do breaks occur?

(Analyze the root causes of potential breaks, such as system errors, data quality issues, or process failures)

5. Root Cause Analysis

What are the root causes of these potential breaks?

(List the components, systems, or processes in your data pipeline that have the potential to fail)

Continuous Feedback Loop

Step 1: Understand Data Quality Issues

Document data quality issues as they occur in production.

Step 2: Immediate Impact Insight

Get immediate insight into the impact of each issue.

Step 3: Pinpointing Breaks

Pinpoint the spots along the data pipeline where something broke.

Step 4: Taking Action

Take action to fix the issue.

Step 5: Learning and Prevention

Learn from the issue so it (and similar issues) don’t reoccur.

Ready to build trust in your data across your organization?

Explore more about data observability with Bigeye by visiting our blog, reading our case studies, or requesting a demo today.

Request a demo
Talk to someone from our team to take Bigeye's data quality monitoring tool for a spin.

Check out the Bigeye blog
Get expert insights, self-assessments, interviews, and long-form guides on the latest and greatest in data.

Read Our Customer Stories
Learn from the experiences of others and gain insights into how Bigeye can help your organization achieve data excellence.

Bigeye