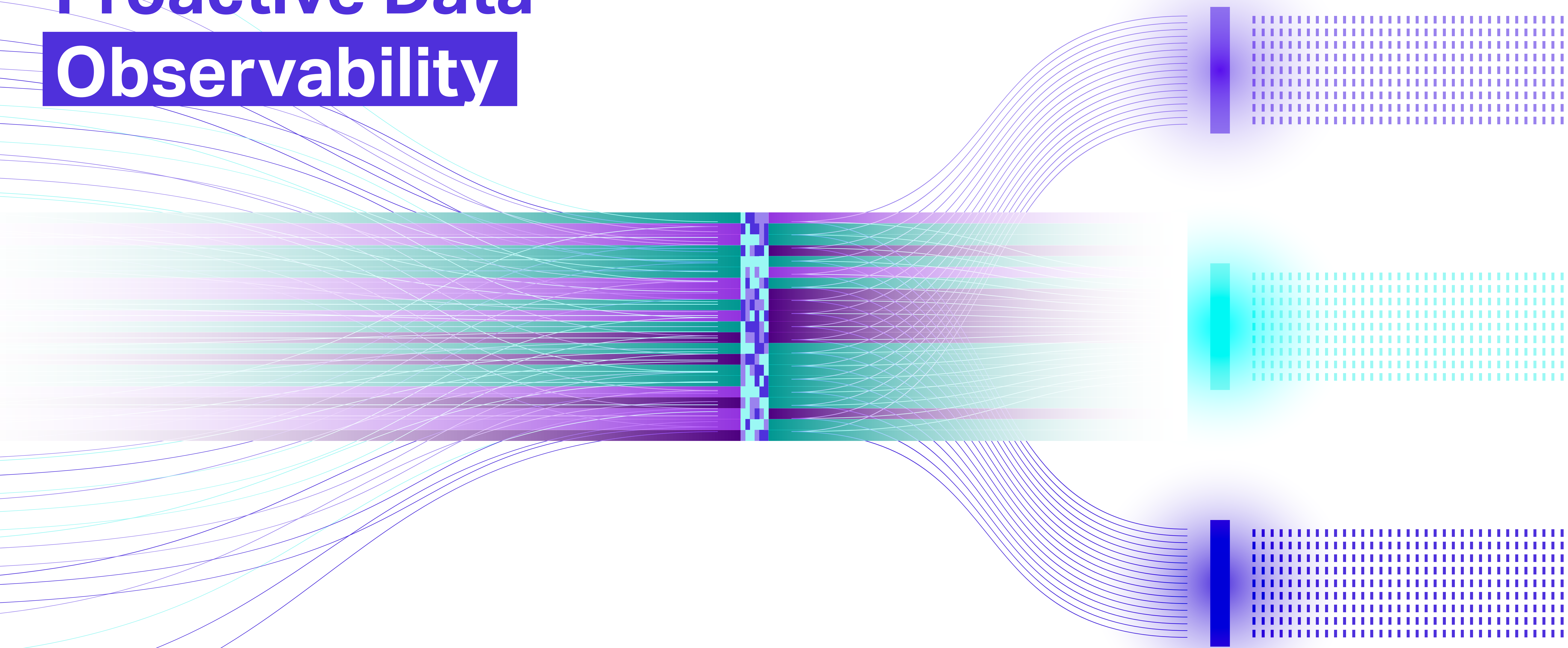
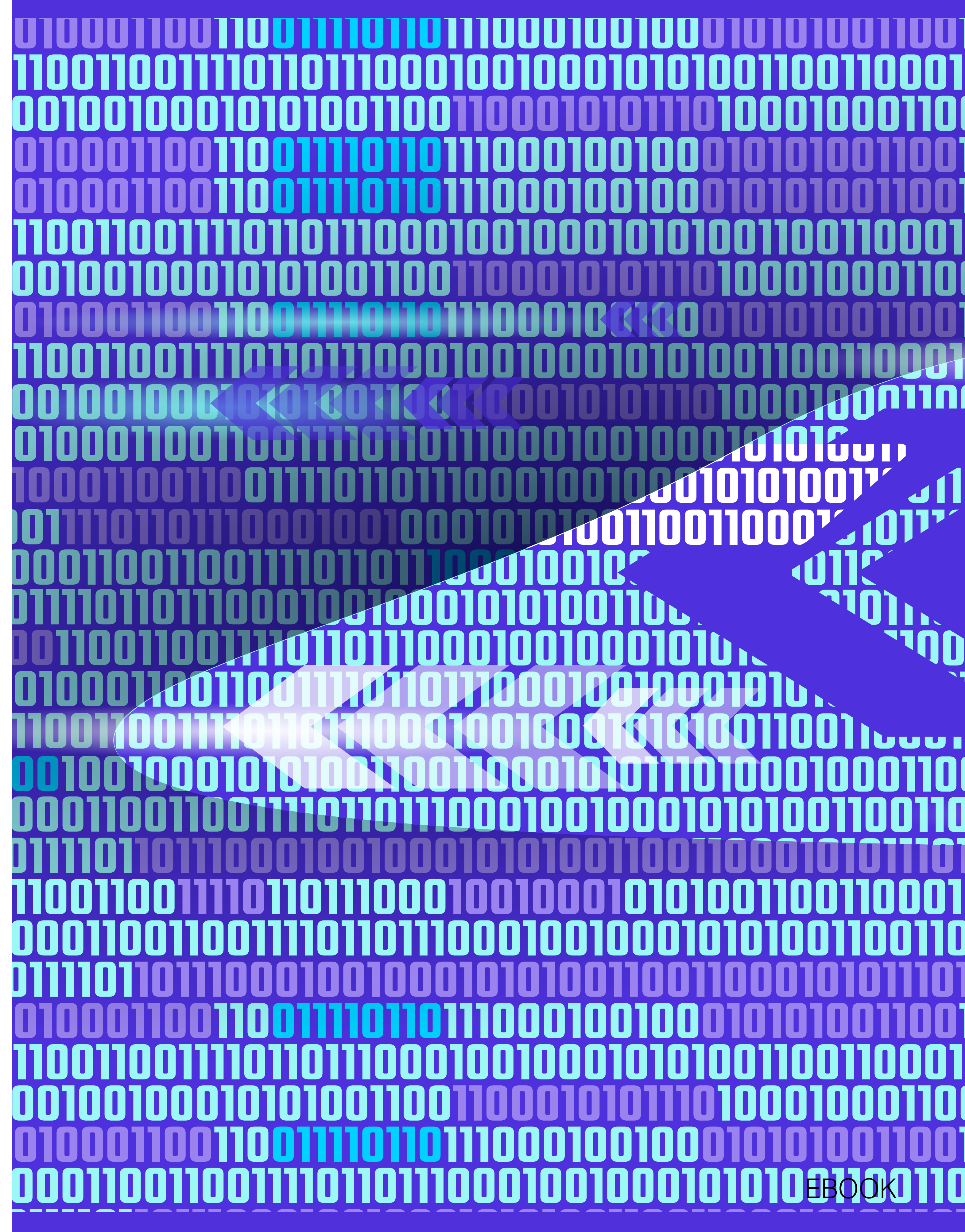


# 5 Steps to Achieve Proactive Data Observability





- 1 What is Data Observability?
- 2 Challenges on the Path to Data Observability
- 3 5 Steps to Achieve Proactive Data Observability
- 4 Why Proactive Data Observability Matters



# What is Data Observability?

Data runs the world. Seriously. It's estimated that we create 2.5 quintillion bytes of data each day. That's 18 zeros.

This influx of data has created challenges for organizations whose infrastructure relies on data, largely because they weren't fully prepared for today's volume of data, the variety of data sources, and the complex transformations all that data requires.

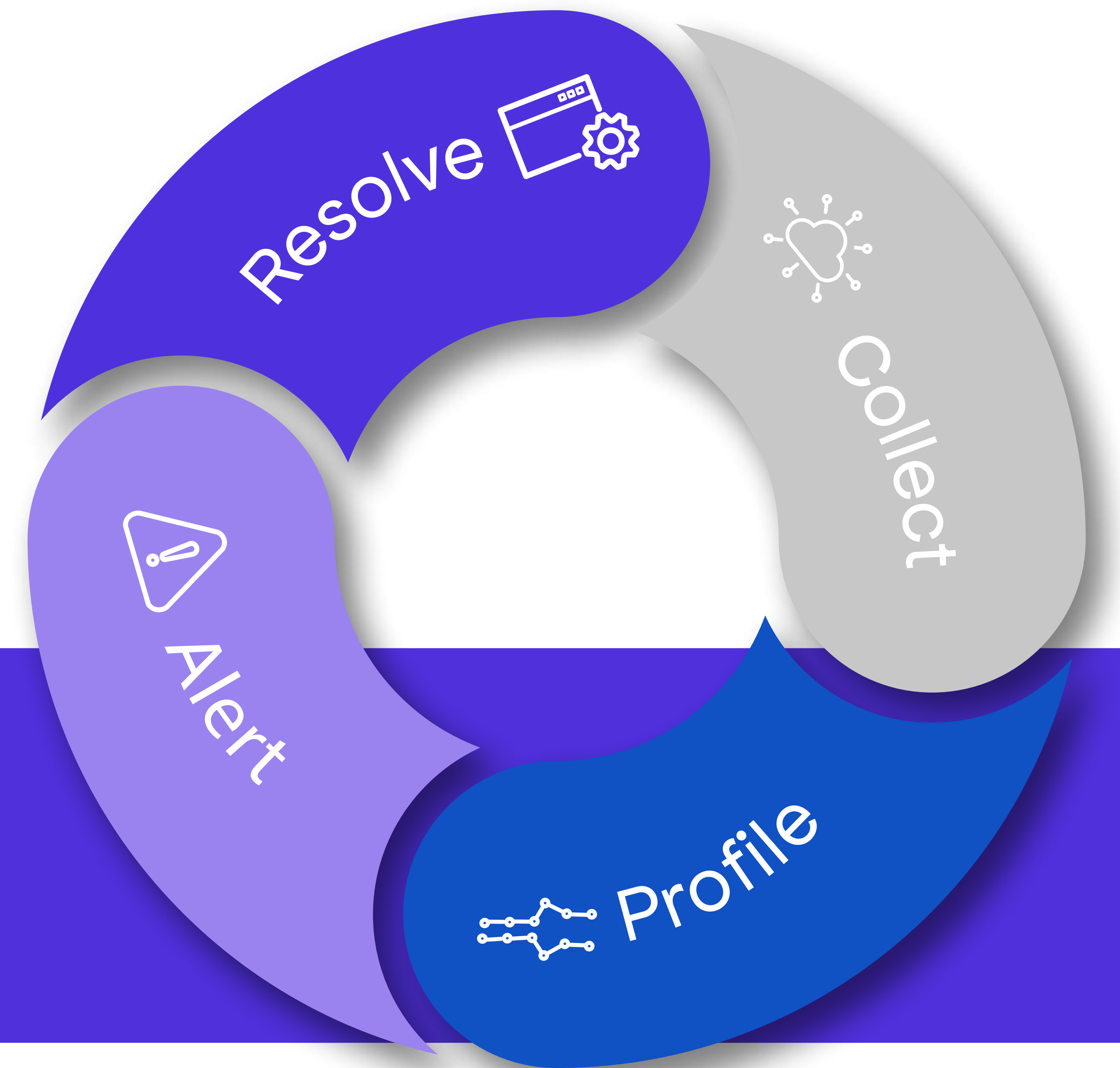
On top of that, there's been too much of a focus on analytics engineering over data engineering – the latter of which is essential to making sure data quality is in a good place to power those advanced analytics.

“Data pipelines are growing in size, volume and complexity, with multistage processing and dependencies between various data assets. Today, data engineering comprises 80% to 90% of the work organizations do with data.”

*Gartner - Data Engineering Essentials, Patterns and Best Practices | May 2021*

When so much hinges on having good data in place, a lot can go wrong in a highly visible way. Just ask any data or engineering team that's received a frantic call from the CEO. Enter [data observability](#).

Data observability is about understanding your system's health and state of data. It relies on several activities and technologies to enable teams to collect, profile, alert, and resolve data issues in near real-time.





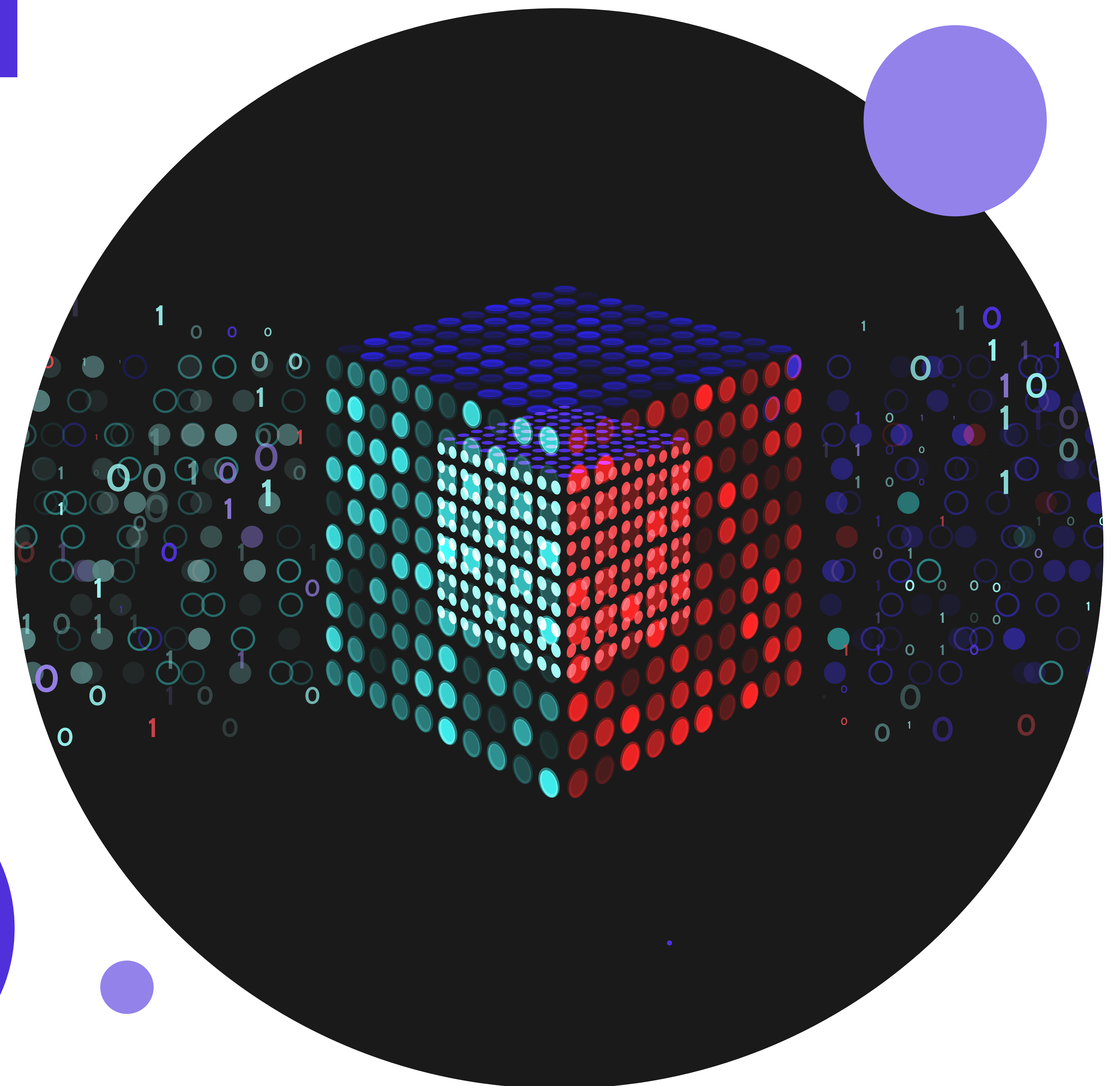
# What is Data Observability?

In practice, data observability makes Data Operations (aka DataOps) possible. It does so by not just shedding light on a problem but also by providing the context data engineers need to solve the problem and understand how to prevent it from recurring.

The key to making this happen is to ensure data observability is infused consistently throughout the end-to-end DataOps workflow. That way, all of the activities involved are standardized and centralized across teams for a clear and uninterrupted view of issues and impacts across the organization.

Of course, achieving data observability is easier said than done (because isn't everything?). But it's not impossible by any stretch – it simply requires the right approach.

This eBook will explore that approach in detail, including the common challenges organizations face along the way and five steps to overcome those challenges to achieve proactive data observability.



# Challenges on the Path to Data Observability: It All Starts with Monitoring

Data observability encompasses a series of activities, including monitoring, alerting, tracking, comparisons, analysis, logging, and SLA tracking.

While [data monitoring](#) is only the first piece of this equation, it's one of the most important – because, without the awareness that it provides, there's no way to identify and take action on issues. Otherwise, data is just a black box.

So what exactly does this data monitoring entail? It's the ongoing process of measuring your data's fitness for use, and it requires a deep knowledge of what's going on in the data pipelines. Importantly, it provides insight into what issues are occurring and what the source is and what the impacts could be elsewhere.

Data monitoring might sound simple enough, but in practice, it's ripe with challenges that can cascade down to the other elements of data observability – stopping the train before it ever really leaves the station. Two of the most critical challenges are the technology at play and the people involved.



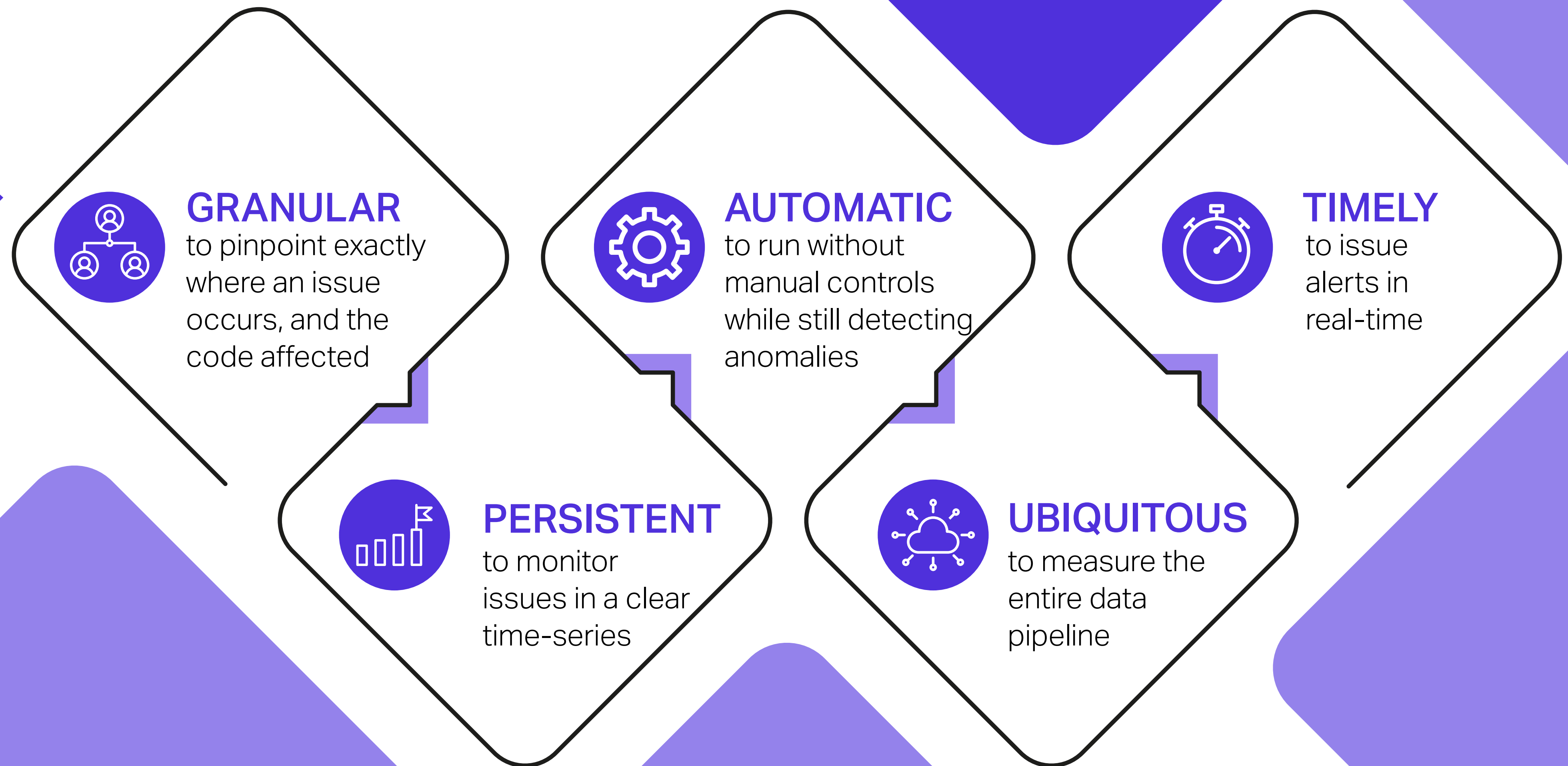


# Technology: How data monitoring gets handled

When it comes to effective data monitoring, traditional application performance management tools won't cut it. They're simply not granular enough to get the right level of detail.

Instead, teams need dedicated data monitoring software that's designed specifically to be:

Only when armed with these details can data engineers properly monitor data's fitness for use and identify and take action on issues that might arise.



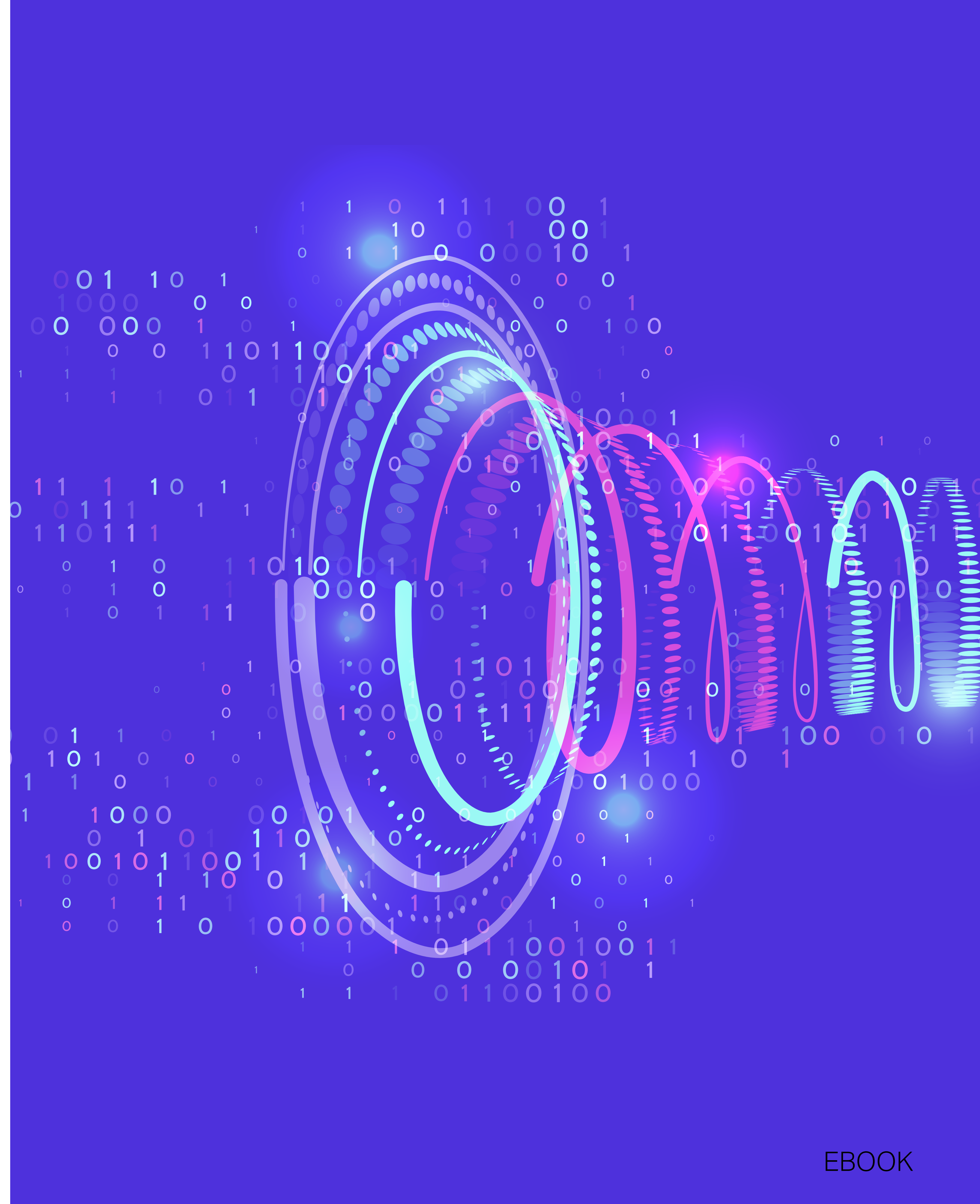
# Team: Who gets involved and relies on data pipelines

More data means more teams using that data in new ways, including (or rather, especially) non-technical teams. Everyone from data scientists to executives now needs data to do their jobs, and this has brought more people – one might argue too many people – into data pipeline conversations.

The more involved people, particularly those who aren't necessarily well versed in the nuances of a complex data pipeline, the more challenging data projects become. Ultimately, people speak different languages and have competing ideas and priorities, leading to less than ideal results.

Perhaps most alarmingly, they fail to involve data engineers early on when their expertise is needed most. The earlier data engineers get involved, the better positioned they are to help set requirements.

Additionally, any fixes to the data pipeline are cheaper to make sooner than later.

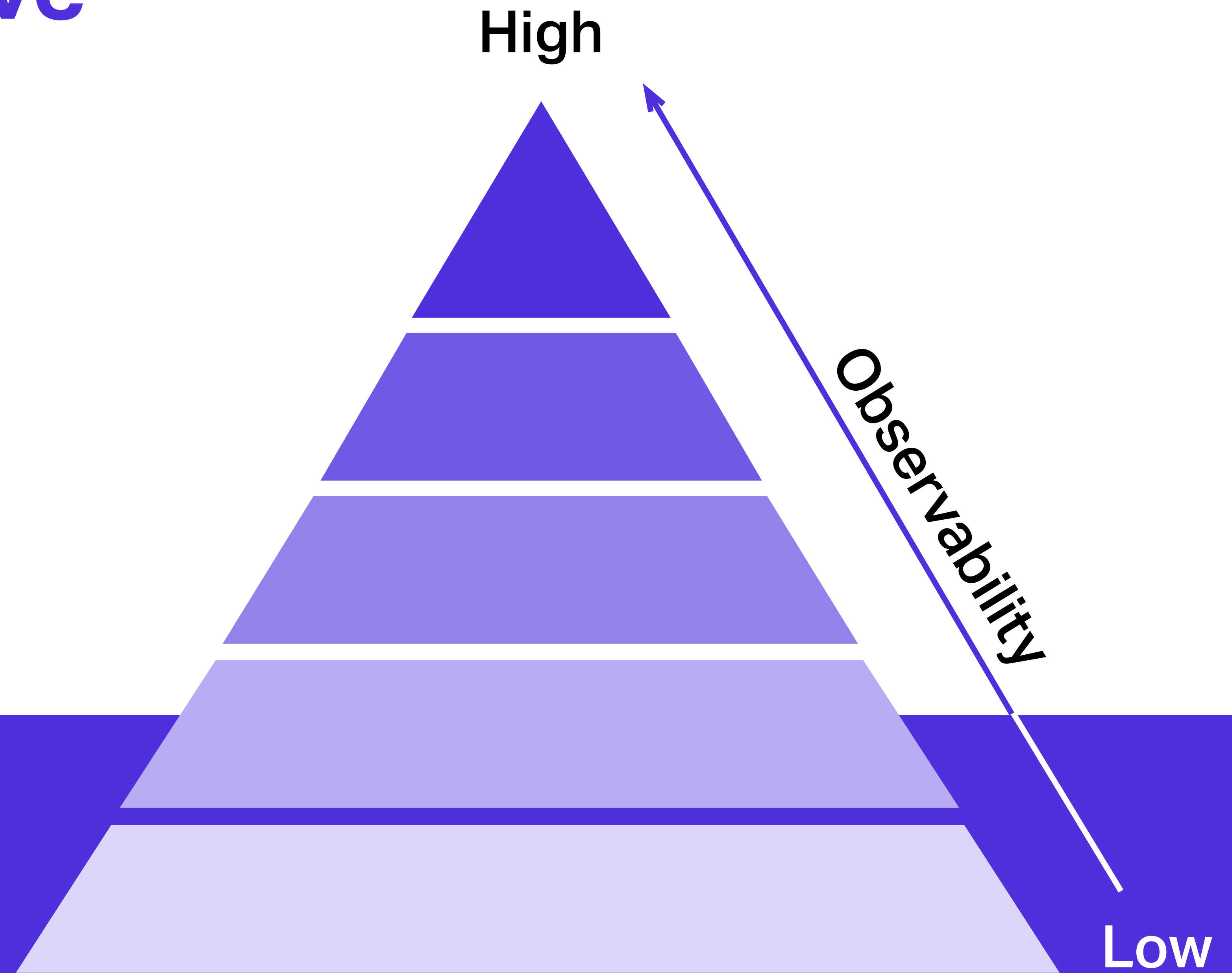




# 5 Steps to Achieve Proactive Data Observability


When it comes to getting data observability right, the holy grail is being proactive. This proactive approach makes it possible to get ahead of issues rather than constantly working in break-fix mode.

But where do you go from there? We've put together these five steps that can help lead the way. And for each step we're going to use the analogy of a cargo train to paint a clear picture of how to measure your observability success.





# Step 1 Understand Pipeline Execution Health

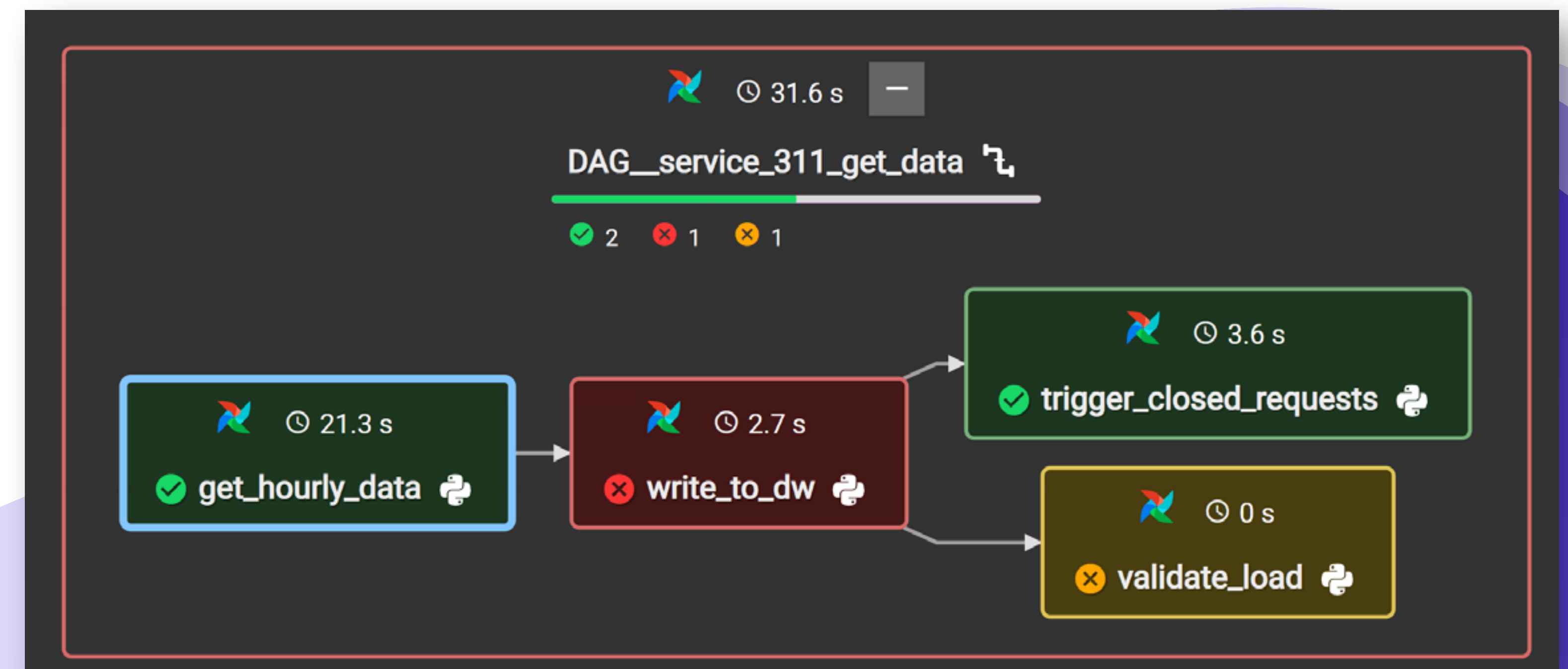
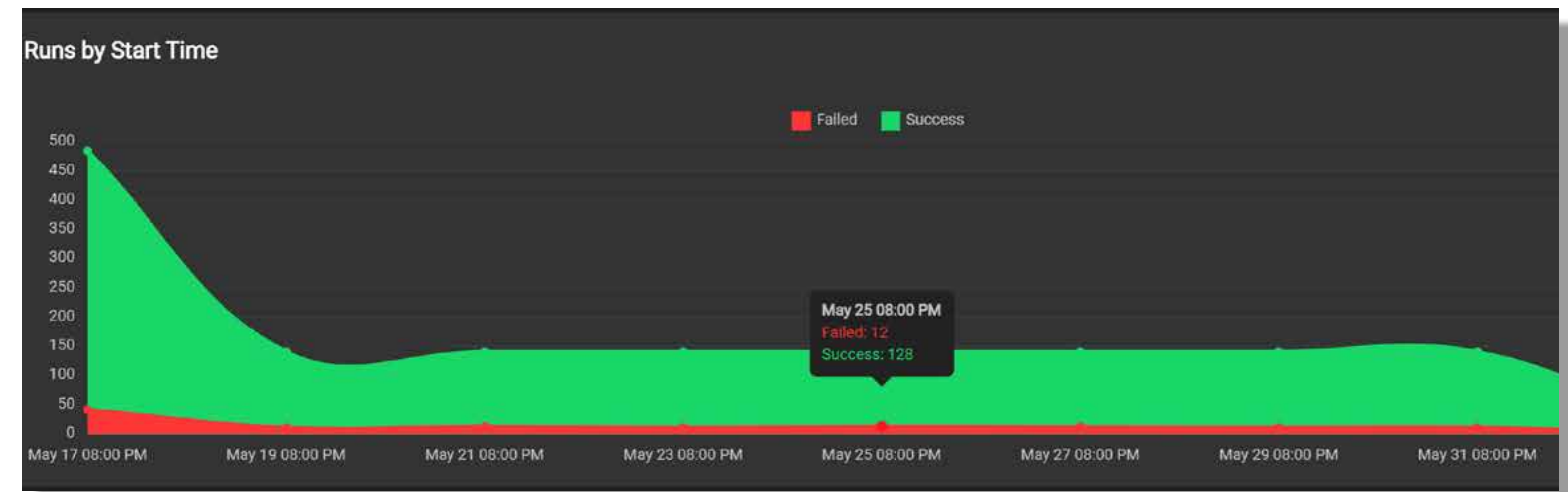
 Has your cargo train left the station? If not, why hasn't it? Is it broken down or just not moving?

The very first thing to understand is whether or not your data pipeline is executing correctly, as this impacts the flow of data. Specifically, a data orchestrator like Airflow or Dagster will help create pipelines to run data through, but how do you know whether the data is moving through them as planned?

If the pipeline fails, then no data can actually come through, and everything else that follows won't execute correctly.


This dashboard offers a clear way to visualize pipeline health and ensure your pipelines are executing as expected. If they're not, it puts a full stop to all data reliability questions that follow, so having a quick and easy way to understand pipeline health is critical.

Beyond providing a clear view of pipeline health, this dashboard also makes it possible to quickly isolate where the issues are occurring so teams can jump into fix mode right away.





# Step 2 Alert on Pipeline Latency

 Did your train arrive on time?  
Or did it arrive too early, too late, or not at all?

Pipeline	Failures Rate ↓	Failures Count	Errors Count	Runs Count	Average Run Duration	Total Run Duration
<a href="#">daily_sales_ingestion</a> Global Sales databand-internal-sa-demo-af	25 %	14	11	56	21.2 s	19 m, 46.2 s
<a href="#">service_311_get_data</a> Service 311 databand-internal-sa-demo-af	10 %	31	24	322	55 m, 32 s	1 w, 5 d, 9 h, 59 m, 26 s

Next, it's important to determine if data arrives in a usable time window based on when it's expected to come through. For example, a company expects to have data in hand for users at a certain time for them to make critical business decisions.

Any data that arrives late or not at all creates serious problems.

As a result, pipeline latency – or the amount of time it should take data to arrive from point A to point B – is essential to nail down and maintain on an ongoing basis.

Monitoring pipeline stats like run count, average run duration, and total run duration can help address pipeline latency issues by keeping tabs on how often a specific process runs and how long that run actually takes versus the average run.

Comparing the two side-by-side can ensure the expected and the actual runtime align. If they don't, it's a clear indication of an issue that requires further attention.

Run list | Metrics | Alerts **110** | 5 Alerts Defined


31 Alerts | Acknowledge | Resolve

	Severity	Description	Trigger Value	Origin
<input type="checkbox"/>	HIGH	Missing data operation: service_311_get_data	2	Multiple datasets service_311_get_data scheduled_2022-06-03T09:00:00+00:00
<input type="checkbox"/>	CRITICAL	Run failed: service_311_get_data	failed	service_311_get_data scheduled_2022-06-03T09:00:00+00:00
<input type="checkbox"/>	LOW	Distinct count > 5 in [borough] column	6	service_311_get_data scheduled_2022-06-03T05:00:00+00:00
<input type="checkbox"/>	MEDIUM	Distinct count > 5 in [borough] column	6	NYC 311 API service_311_get_data scheduled_2022-06-03T05:00:00+00:00



# Step 3

## Check Data Sanity

 What cargo was on your train when it arrived? Was it the same cargo you put on and expected, or did you end up with more or less?

Once the data arrives, does it come through valid and complete, or are there any errors? This data sanity check is critical because even if data arrives on time, it doesn't do any good if it's rife with errors.

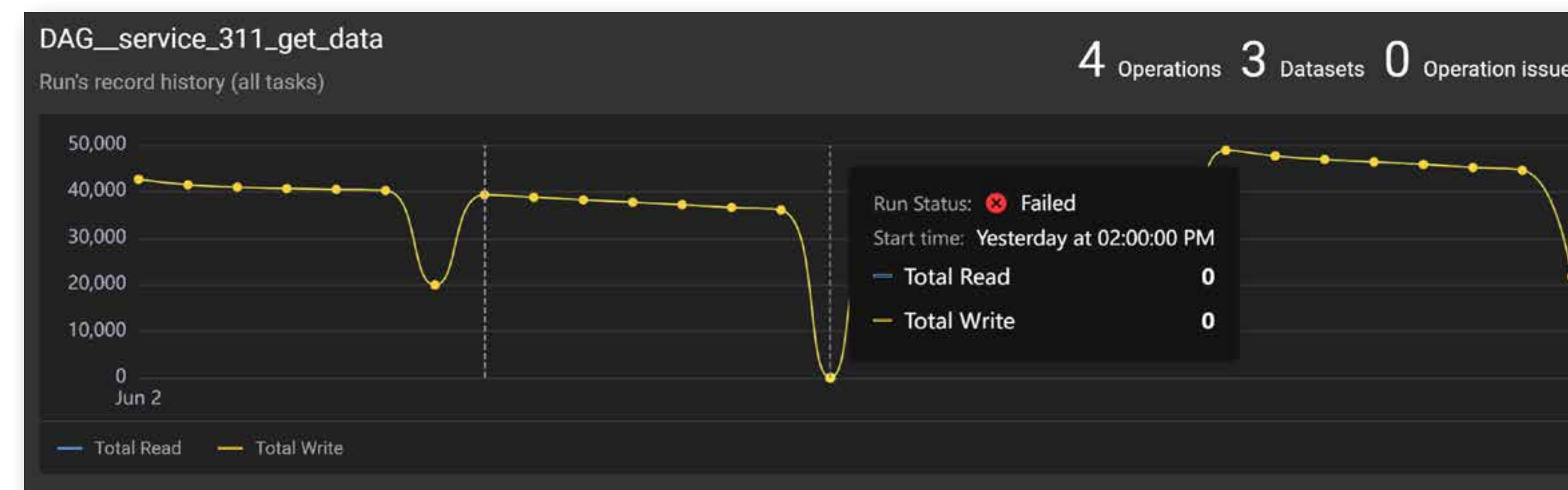
Data sanity centers around any changes to the data schema, such as whether or not there are more or less columns than expected.

Checking data sanity at scale (think millions of data points moving through the pipelines) can get very difficult, requiring the right team and dedicated technology.

Checking data sanity requires digging deeper into the data schema for any given run. A dashboard like this provides an easy way to pull up each data schema and get a bird's eye view of the columns.

Even better, the ability to click into each one allows for a closer look at the data contained within to look for anomalies.

At the same time, the graphical view provides a quick overview of what's considered "normal" and where anything might fall outside that range.



4 operations


Operations	Schema	Records	Task
Read	42 columns	19,658 records	write_to_dw
Write	42 columns	19,658 records	write_to_dw
Write	42 columns	19,658 records	get_hourly_data
Read	40 columns	19,658 records	get_hourly_data

Run Status: ✅ Success  
Table name: Redshift - 311 Staging Data  
Start time: Yesterday at 07:00:00 AM  
Writing records: 19.658K



# Step 4

## Analyze Data Trends

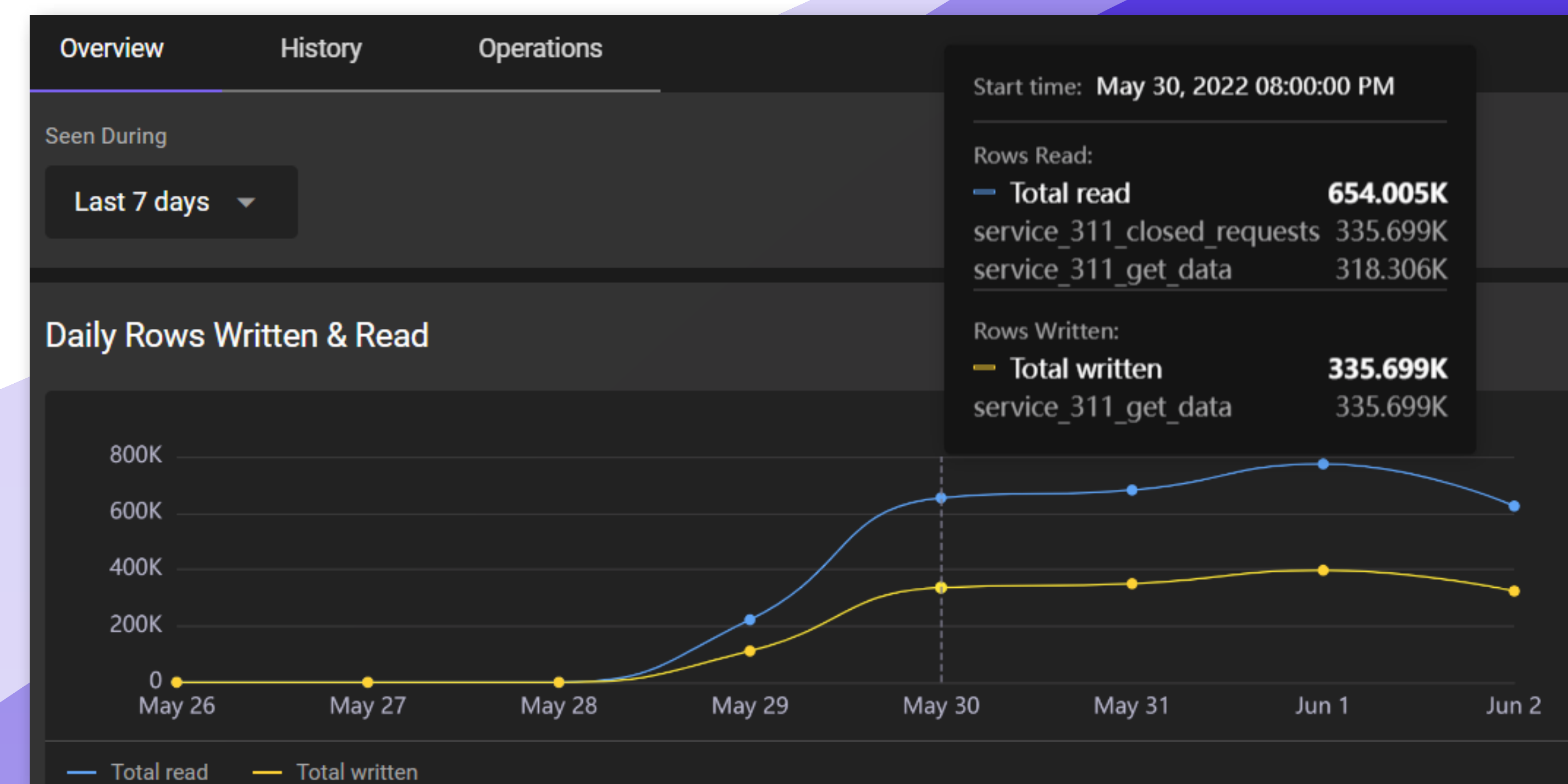
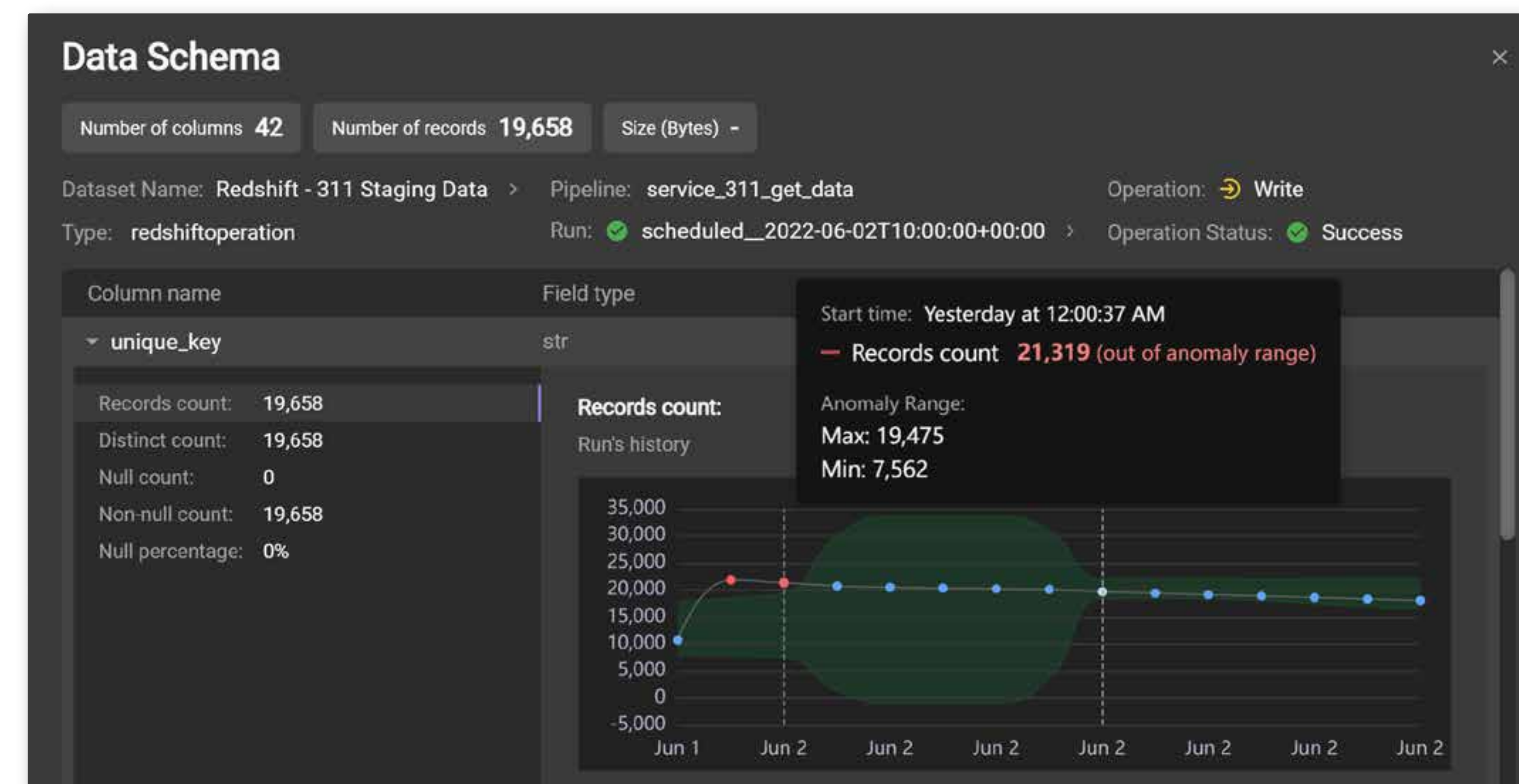
 Did any of the cargo on your train change during transit? Is it on the train as expected, but something about it is different than when it got on the train?

While data sanity looks at the data schema to confirm the data coming in is structurally okay, there's still more to consider. From there, it's essential to dig into the dataset trends to determine what's going on that causes these changes to keep happening.

For instance, what if one of the columns is structurally right, but its data is wildly different from expected?


Narrowing in on changes to the data can help determine what within it is actually changing, including how any structural changes might impact the health of the data over time.

This ability to look closely at not just how the data is changing but what within it is changing, is important to understanding where issues are occurring and what needs to be corrected.





# Step 5 Identify Data Access and Business Impacts

 What's the final destination for the cargo on your train? How is it going to get there once the train stops, and how will any issues with the train itself impact its final arrival?

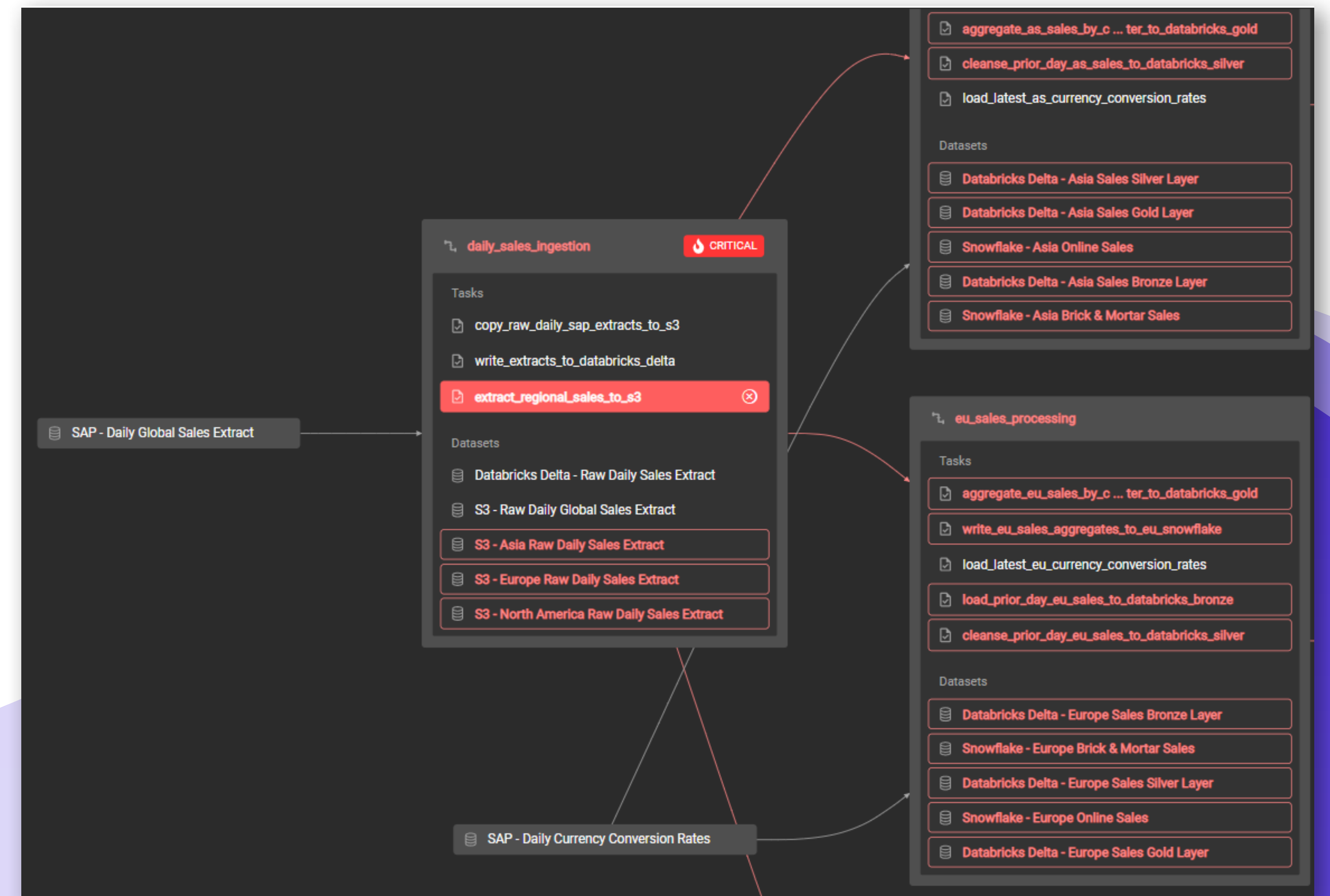
The last step brings everything together to identify the business impact of issues at any point by understanding how the data maps to key decisions.

Gaining this understanding requires investigating how the business uses each set of data.

That way, it's easy to identify the downstream impacts of any issues in terms of latency, sanity, trends, or anything else.

A quick way to understand the data flow, for example, by looking at lineage, can help identify the end destination – and, therefore, the business impacts.

It's important to look closely at this view and related alerts to understand the different data sets that will be affected by any issues associated with a particular pipeline or data run.



# Why Proactive Data Observability Matters

The importance of data observability can't be underestimated: It allows data teams to understand what's happening with their tools and infrastructure to identify errors down to the source and determine the impact of those errors quickly.

In doing so, data observability allows for greater agility and iteration. And what organization doesn't want that?

Critically, this data observability must be proactive. The five steps outlined here – understanding pipeline health, pipeline latency, data sanity, data trends, and business impacts – will help achieve that level of proactivity. But those alone are not enough.

Every team must also approach each of these steps in a standardized and centralized way, and that's where having a dedicated solution that can provide the appropriate level of monitoring and analysis comes into play.

With that combination of process and technology, your team will be well positioned to achieve proactive data observability and – as a result – improve agility and iterate faster.

## Data Access

How do these issues map to business decisions?

## Data Sanity

Is the data coming through valid and complete? Are there errors in the data itself?

## Pipeline Execution

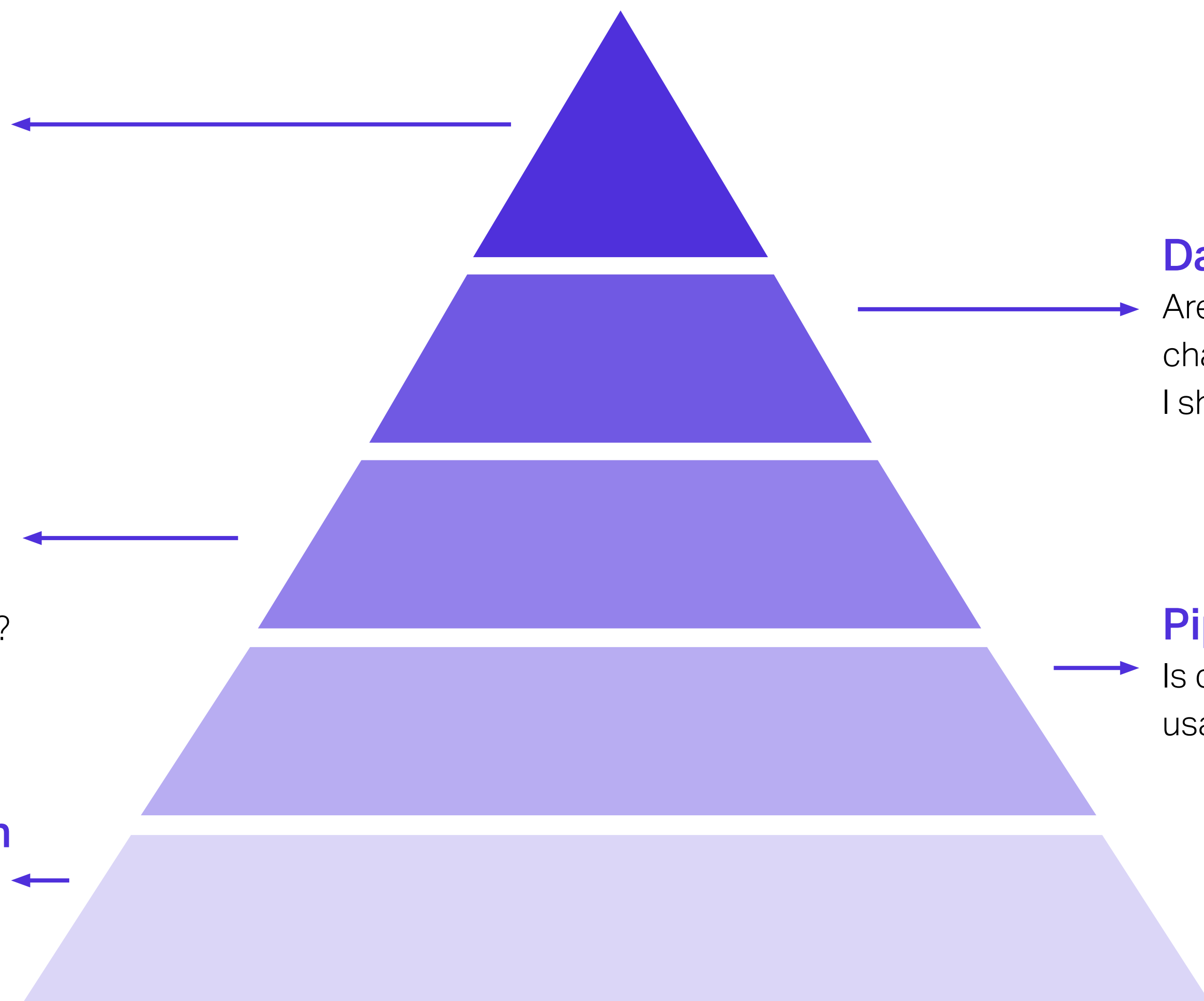
Is data flowing through my pipelines?

## Data Trends

Are there important changes in the data that I should know about?

## Pipeline Latency

Is data arriving in a usable window of time?





# Wrapping it up

Implement proactive data observability today and know when there's a data health issue before your consumers do.

Discover how you can get alerts on leading indicators of data pipeline health issues and drill deep into the data to implement a fix before bad data gets through.

[Request a Demo](#)

