Qualibri Tech

✳ Case study

# Their Tests Passed. The Platform Still Had 270 Defects.

How a structured data quality assessment exposed hidden risks, strengthened trust in critical data flows, and helped a Snowflake + dbt platform reach production on time.

| | | |
|---|---|---|
| **Client** | → | European pharmaceutical and healthcare company |
| **Stack** | → | Snowflake × dbt × Python × AWS × REST APIs × Azure DevOps CI/CD |
| **Challenge** | → | New platform, aggressive release timeline, a previous quality failure, and no clear view of whether the critical data flows could actually be trusted. |
| **Solution** | → | Quality assessment × risk-based testing strategy × expert validation × code-based automation on the existing stack |
| **Result** | → | 270+ defects found before production × ~80% increase in unit test coverage × on-time release × only 5 defects in Hypercare |

## Client profile

The client is a European pharmaceutical and healthcare company. Its new data platform feeds decisions about ongoing clinical studies — the kind where a data error can put patients at risk. Data flows in from multiple external providers, is consolidated and enriched, and flows back out to partner systems that act on it.

For most data teams, the equivalents are a customer-facing dashboard, a revenue report the board reads, or an AI feature. Data trust is identical: data leaves your platform, someone believes it, and you can't undo that.

Under the hood, the platform runs a familiar stack: Snowflake consolidates the data, dbt builds the warehouse structure and transformation logic, AWS orchestrates ingestion, and REST APIs handle delivery in both directions.

## The challenge

The request looked routine at first: validate the new warehouse before release, in about 3 months. The previous release had failed on quality, so this one was non-negotiable.

Qualibri Tech started with a structured data quality assessment that found risks the client's team could not see from inside:

- **Business logic wasn't protected.** Existing tests focused on schemas, nulls, and duplicates rather than the rules that determined whether downstream data could be trusted — the layer where most defects were later found. Ingestion and outbound data flows had never been systematically verified.
- **New architecture, old tests.** Documentation was incomplete and partly outdated. The architecture had been rebuilt as a layer-based Data Vault, but test coverage had not moved with it.
- **No unit testing existed.** The in-house automation engine ran Excel-defined tests converted to YAML, stopped on any failure, and took about a week per cycle.
- **Defects had no owner or process.** No triage, no link between defects and test cases, and a test team working in isolation from developers.

Every signal the team watched was green, which is exactly why nobody saw a problem. And nobody could confidently answer one question: "Can we trust the data the next release will produce?"

The assessment gave the client more visibility into what was covered, what was under-tested, and what to fix first, ordered by business risk.

# The solution

The biggest risk wasn't failing tests. It was successfully testing the wrong things. So the strategy was built on the assessment's map, not uniform execution:

- **Sequence testing by business risk.** Validation started with ingestion, then transformation logic, then outbound delivery — the same sequence in which incorrect data would affect the business. About 260 of the defects surfaced inside the warehouse — in the transformation layers between raw, staging, and the business data model.
- **Test the platform's safeguards.** Controlled negative tests replaced valid data with invalid data — nulls, duplicates, broken referential integrity — to verify that the platform would stop the load instead of passing bad records downstream. Often, it did not.
- **Structure at scale.** Roughly 1,000 test cases, grouped and parameterized, ran in up to seven parallel streams. Weekly triage linked every defect to its test cases and grouped fixes into a prioritized list for developers. This turned hundreds of individual issues into a prioritized remediation roadmap instead of an unmanageable defect backlog.
- **Code-based automation on the existing stack.** A proof of concept replaced the in-house engine with Python and dbt tests reporting into Azure DevOps: 5 parameterized tests expanded into 100 executed test cases, completing in under 5 minutes — work that had previously taken about a week. While a broader rollout was not implemented within the engagement, the POC established the path. No new tool was bought.

# Results & metrics

| Metric | Before | After |
|---|---|---|
| Defects in production | Previous release failed on quality | **270+ found and fixed pre-release; only 5 in Hypercare** |
| Unit test coverage | No unit testing framework | **Up ~80%, via dbt tests across transformation layers** |
| Automation cycle | ~1 week per 100-test run; halted on any failure | **Under 5 minutes for 100 tests (proof of concept)** |
| Critical data flows | Ingestion and extraction untested | **Coverage extended across the flow — ingestion, warehouse transformations, and outbound delivery** |
| Release | At risk | **Shipped on schedule; regulatory validation approved** |

Finding 270+ defects was critical. Yet, the real outcome was knowing exactly where the platform was vulnerable — and fixing those risks before production.

The majority of the found defects were major or critical issues in how the platform processed, moved, and preserved data — each one detected before a single production record was touched. Besides, the ~80% increase in unit test coverage has become a lasting asset: the client's own team now runs dbt tests on every change.

The resulting evidence also passed independent regulatory validation, demonstrating that the testing approach was robust enough not only for engineering teams but also for external auditors. If the coverage holds up there, it holds up in front of your board.

## Wondering what you should actually be testing?

Start with a 30-minute scoping call. We will tell you which flows we would map first and why — and honestly, whether an assessment is worth it for your platform.

## Contact me to book the scoping call

**Natalia Zhornyk**

Founder
at Qualibri Tech