Qualibri Tech

✳ Case study

The Data Ran the Business. Nobody Was Testing It.

How Qualibri Tech built a data-quality capability from scratch for a complex iGaming & iLottery platform, cutting release-blocking defects 90% and regression from 5–7 days to under an hour.

Client	→	A global iGaming & iLottery technology platform
Stack	→	Snowflake × dbt × Python × Airflow × Azure DevOps CI/CD
Architecture	→	Shift from legacy MS SQL Server → Data Lake → Data Vault
Challenge	→	Millions of bets an hour, 13 operator clients — and no testing function at all
Solution	→	Built a data-QA capability and code-based automation from scratch, then hardened it through a Data Vault re-architecture
Result	→	**90% fewer release-blocking defects × 790 checks automated × regression: from 5–7 days to <1 hour × 75% fewer production defects on automated flows**

Client profile

The client is a global iGaming & iLottery technology platform. Real players bet real money online 24/7, and the platform takes in millions of bets an hour. Those bets become the reports its operator clients use to decide what to run, what it earns, what tax they owe, and what their regulators need to see. There were 13 such operators across North America and Europe when the project began — today, 20.

The data runs on a legacy Microsoft SQL Server source, a Data Lake, and a modern Snowflake + dbt warehouse, all orchestrated by Airflow. Each bet passes through multiple transaction states, and the money totals have to reconcile. A small transformation error can quietly distort a client's report.

The challenge

When Qualibri Tech joined, the Data & BI organization had no dedicated testing function. Anything built was deployed straight to production. Defects were found by users, not engineers. Reports ran on incomplete or wrong data, escalations were constant, and one operator relationship had come close to termination over quality.

The difficulty went beyond the missing process. Midway through, the architecture itself was rebuilt — from a medallion design to a layer-based Data Vault, with hubs, links, and satellites spreading related data across far more tables. Testing had to keep pace with an architecture changing underneath it: knowing which data belonged in which report, and proving it survived every transformation intact.

The initial ask was simple: bring in a tester and get the defect count down. But the real job was building trust in the data from the ground up.

The solution

The engagement started as manual validation and grew, by design, into a maintainable code-based framework.

First, control. Testing moved onto a UAT environment with a real release cadence, which meant reports were verified before they reached production instead of after. Defect management — until then nobody's job — became an independent, tracked process in Azure DevOps, with weekly triage. A backlog of roughly 180 unmanaged defects, some critical for months, was cleared in about 3 months.

Then, automation on the existing stack. dbt tests covered the hard part: transformation logic, cross-layer verification, row-count reconciliation, source-to-target checks, and referential integrity. A Python framework was integrated with Azure DevOps by Test Case ID; Airflow scheduled the runs and CI wired them into the pipeline. Tests were built as reusable, parameterized macros, so onboarding a new operator no longer meant rewriting the same checks with a different client's name hard-coded in.

Finally, scale. The QA function grew from 1 engineer to a small team, trained on Snowflake, dbt, and SQL, with a structured onboarding plan replacing informal, word-of-mouth knowledge.

Results & metrics

Before	After
No testing function, defects found by users in production	**Structured QA on UAT; release-blocking defects down 90%**
~180 unmanaged defects, some critical for months, no triage	**Independent weekly triage; backlog cleared in ~3 months, ~30% reclassified as change requests**
Full regression took 5–7 days, run manually	**Full regression runs hourly, in under 45 min**
Production defects were frequent and user-reported	**Down 75% on automated flows; 1–2 a month, caught internally**
Validation was 100% manual	**Manual validation effort down 70% — 790 checks automated**
No automated test coverage	**~85% automated coverage on the mature platform**

Other results:

- 790 manual validation checks are fully integrated with Azure DevOps by Test Case ID.
- ~850 repeatable checks automated (790 Python tests and 60 dbt tests), saving roughly 12–15 hours per release cycle.
- Onboarding access cut from 1 week to 2 days; new testers productive in 2 weeks instead of 3.

Testing on the newest, Data Vault-based platform is still maturing alongside the platform itself. It's not yet in production, but quality is being built in from the first layer, not retrofitted after launch.

Not sure what your current tests actually catch?

We suggest starting from a focused data quality risk assessment. Book 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