Vigil DQ
VIGILANT DATA QUALITY

FOR TEAMS WHO MUST PROVE THEIR DATA IS FIT TO USE

AI writes your data-quality rules. Humans stay in charge.

VigilDQ profiles your warehouse, lets an AI agent draft the rules your data actually needs, and structurally refuses to run anything a human hasn't approved — with the masked pipeline and audit trail to prove it.

Arrange a 30-minute demo

0–100
one defensible score, six quality dimensions

100%
of rules approved by a named human — never the AI

0
raw values leave your warehouse — masked at the boundary

2 + 1
engines validated live (Postgres, Databricks) · SQL Server next

THE PROBLEM

Data quality is easy to promise — and hard to prove.

In pharma, banking and insurance you don't just need clean data; you need to **certify** it. Hand-writing quality rules for thousands of tables doesn't scale. Pointing a raw LLM at the job creates a different problem:

An unguarded LLM

Hallucinates columns that don't exist. Slips a write into "read-only" SQL. Approves its own work. Its safety net is a prompt — which is advice, not a control.

VigilDQ's bet

Put the guardrails in the **architecture**, not the prompt. Every AI proposal is checked against the live catalog, screened for writes, dry-run on the real engine — and can only be activated by a named human.

HOW IT WORKS

Six steps from raw table to governed dataset

Connect
encrypted, read-only credentials

Scan
profile in-warehouse; samples masked

Propose
AI drafts rules with rationale

Review
a human approves or rejects

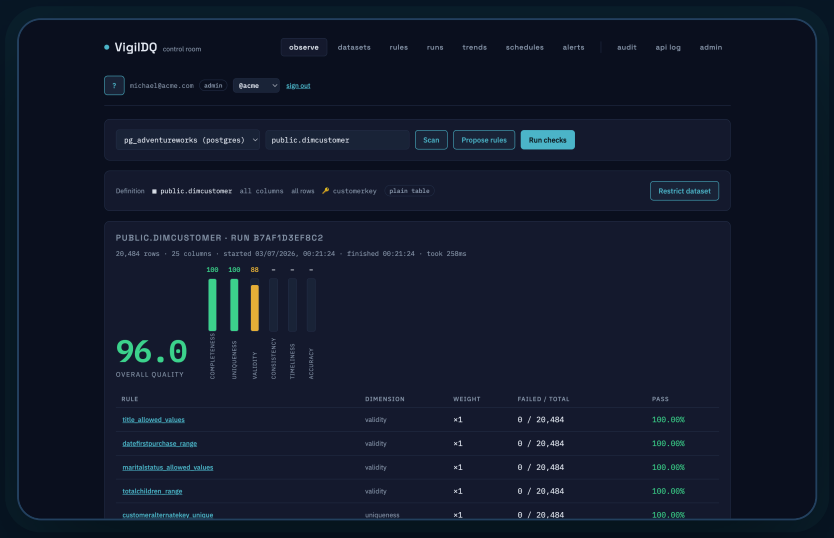
Run
read-only SQL, scored 0–100

Automate
schedules, alerts, pipeline gates

THE PRODUCT, UNRETOUCHED

What your team actually sees

Every screenshot below is the live product, version 0.14.0 — not a mock-up.



One score you can defend

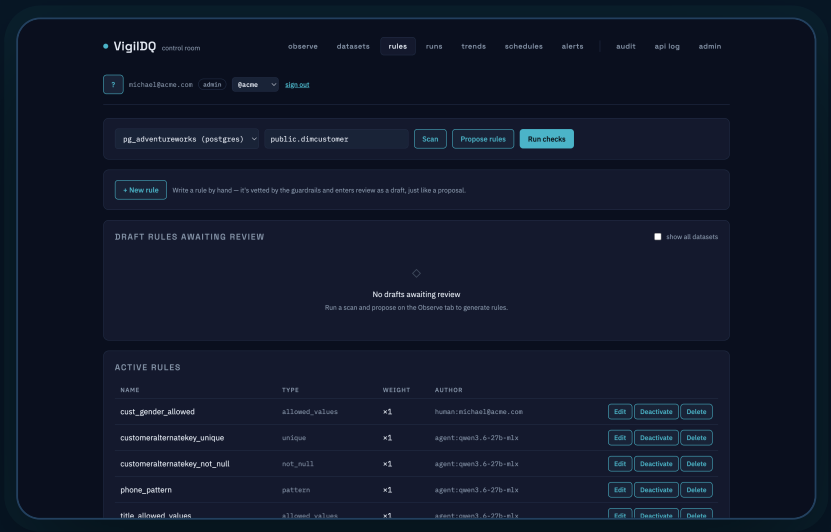
Every run produces a 0–100 scorecard across completeness, uniqueness, validity, consistency, timeliness and accuracy — decomposable to the exact rule and the exact failed count. A rule that errors scores zero, loudly. Nothing is silently skipped.

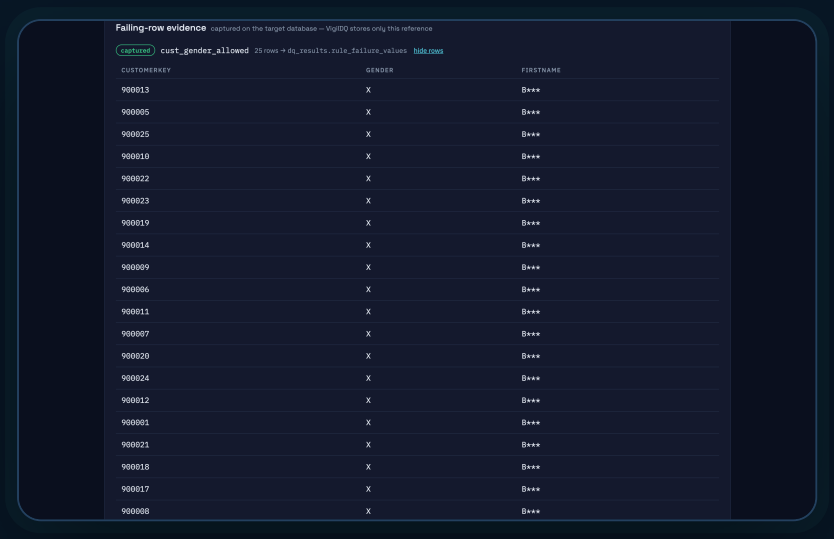
> **WHY IT MATTERS**
>
> A number a steering committee can track, with a drill-down an auditor can defend.

The AI can't approve its own work

Proposals arrive as **drafts**, each with a rationale grounded in your data's profile. Approve, edit or reject — the lifecycle state machine makes AI self-approval impossible, and records the named human behind every activation.

> **WHY IT MATTERS**
>
> Change control is built in, not bolted on: nothing runs without a human's name on it.





Evidence stays in *your* database

Failing rows are captured into a DQ schema on your own warehouse — keys plus offending values, capped and fail-soft. The drill-down masks PII at the boundary (those B*** names are the masking at work).

WHY IT MATTERS

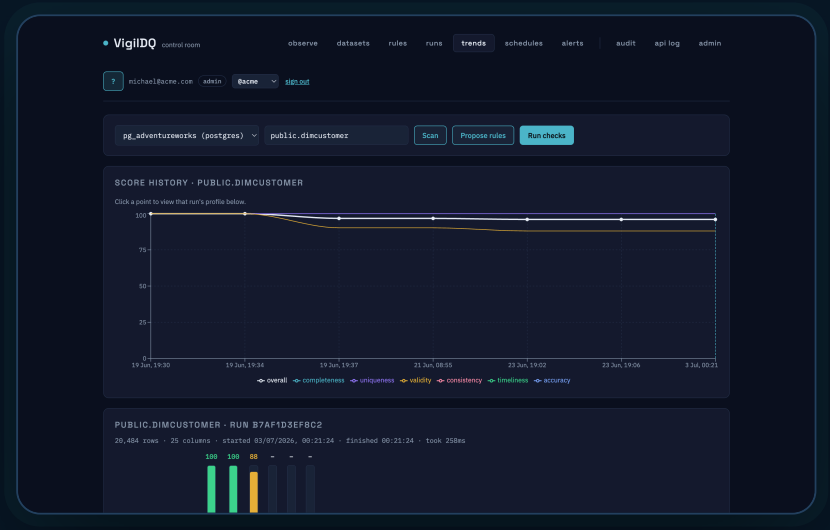
"Show me the failing rows" is an auditor's favourite question. The answer never leaves your estate.

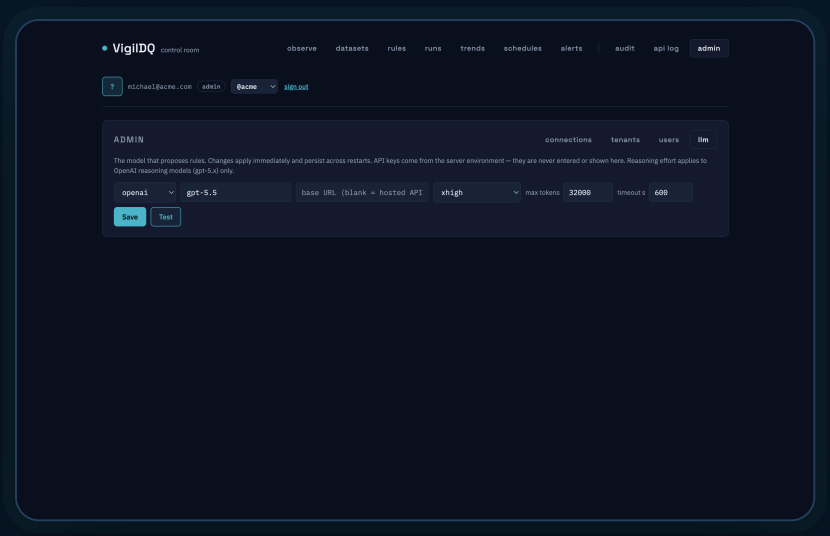
Quality becomes a time series

Every load is scored and recorded. Drift rules — row count, null ratio, schema — judge each run against a median baseline, so a one-off spike doesn't poison the reference and day one raises no false alarms.

WHY IT MATTERS

Regressions show up as a bend in a chart — not as a surprise in production.





Bring your own model — even an offline one

The proposing LLM is a setting, not a dependency: hosted Anthropic or OpenAI (including reasoning models), or any OpenAI-compatible **local** server. Switch it in the admin panel at runtime; every change is audited.

WHY IT MATTERS

With an on-prem model, even the masked profile never leaves your network.

GUARDRAILS, NOT GUIDELINES

Four gates between the AI and your warehouse

SchemaGate

Every referenced column is checked against the live catalog. A hallucinated column cannot pass.

ReadOnlyGate

Lexical DML/DDL screening on top of read-only database sessions. Rules cannot write. Ever.

DryRunGate

Compiled SQL must bind on your real engine before a rule can even be saved — catching what static checks miss.

Lifecycle

Draft → approved → active, and the approval step only accepts a human. The actor comes from authentication, never from a request.

Prompts are advisory. Gates make failure modes impossible. Both have been proven live — catching real attacks, including a ghost column smuggled inside custom SQL.

BUILT FOR REGULATED INDUSTRIES

The boring guarantees that win audits

- **Fail-closed scoring** — an unrunnable rule is a failed control, never a silent pass
- **Provable masking** — tests capture the AI's exact prompt and assert zero raw values
- **Full audit trail** — every change, before/after, with the human behind it
- **Pushdown execution** — queries go to the data; the data never comes out
- **Multi-tenant** — shared platform, isolated tenants, per-tenant roles
- **Complete REST API** — everything the dashboard does, your orchestrator can do
- **Pipeline gates** — pass/fail verdicts Airflow or dbt can branch on
- **0.14.0** — 199 automated tests · Postgres + Databricks validated live

Watch it catch bad data — live.

Thirty minutes on our demo warehouse — realistic, deliberately dirty data — and you watch the agent propose rules against a real schema, a guardrail refuse a bad rule live, and a quality gate stop a bad load.

Arrange a demo **Ask a question**

Nothing to install · no access to your systems needed · the demo runs entirely on our sandbox