VigilDQ — a guided tour

AI-native data quality with guardrails built into the architecture

A screenshot walkthrough of the real product — read it in ten minutes, no meeting required. Every screen shown is live software, not a mock-up.

Version 0.15.0 · PostgreSQL + Databricks validated live · Built for regulated industries

Executive summary

The idea

- An LLM drafts data-quality rules from a profile of your data.
- Hard structural guardrails vet every proposal — then a human approves.
- Approved rules run as read-only SQL inside your warehouse.

What makes it different

- Guardrails are architecture, not prompt instructions.
- The AI can never approve its own work — the lifecycle forbids it.
- PII is masked before the AI or the browser ever sees a value.

Where it stands

- PostgreSQL and Databricks / Unity Catalog validated live.
- 199 automated tests; full audit trail; multi-tenant service.
- SQL Server dialect implemented, live validation next.

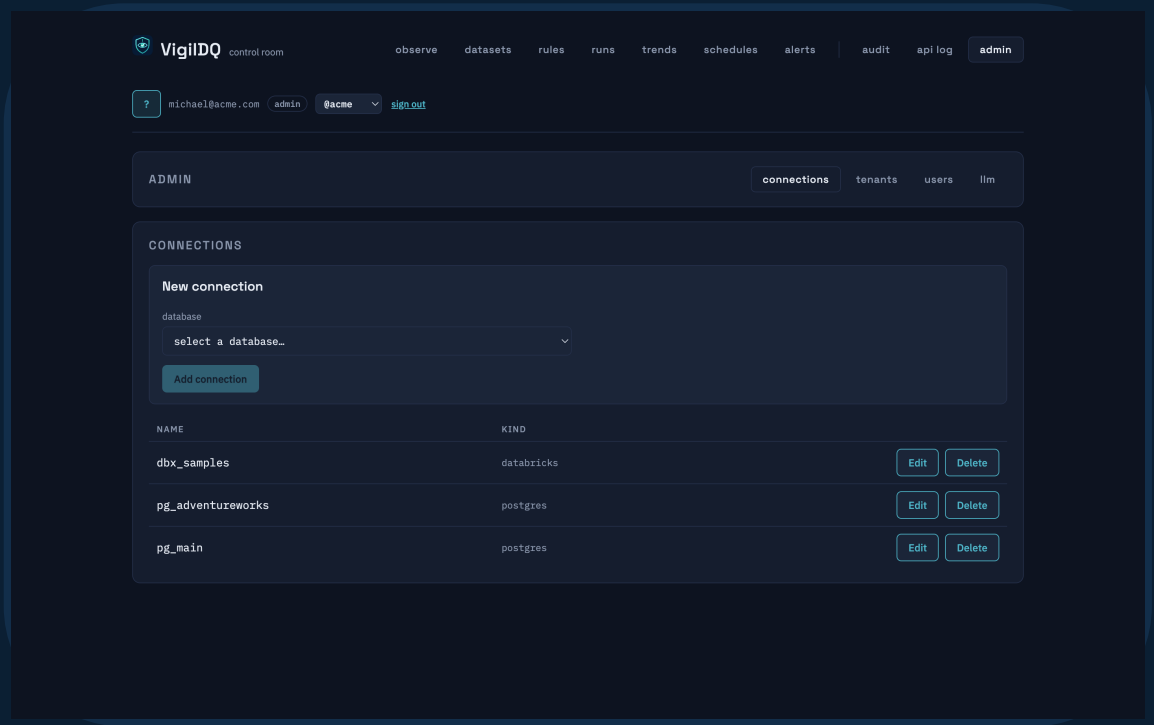
Who it's for

- Pharma, banking, insurance — teams that must PROVE data quality.
- Data platform owners who want AI speed without AI risk.

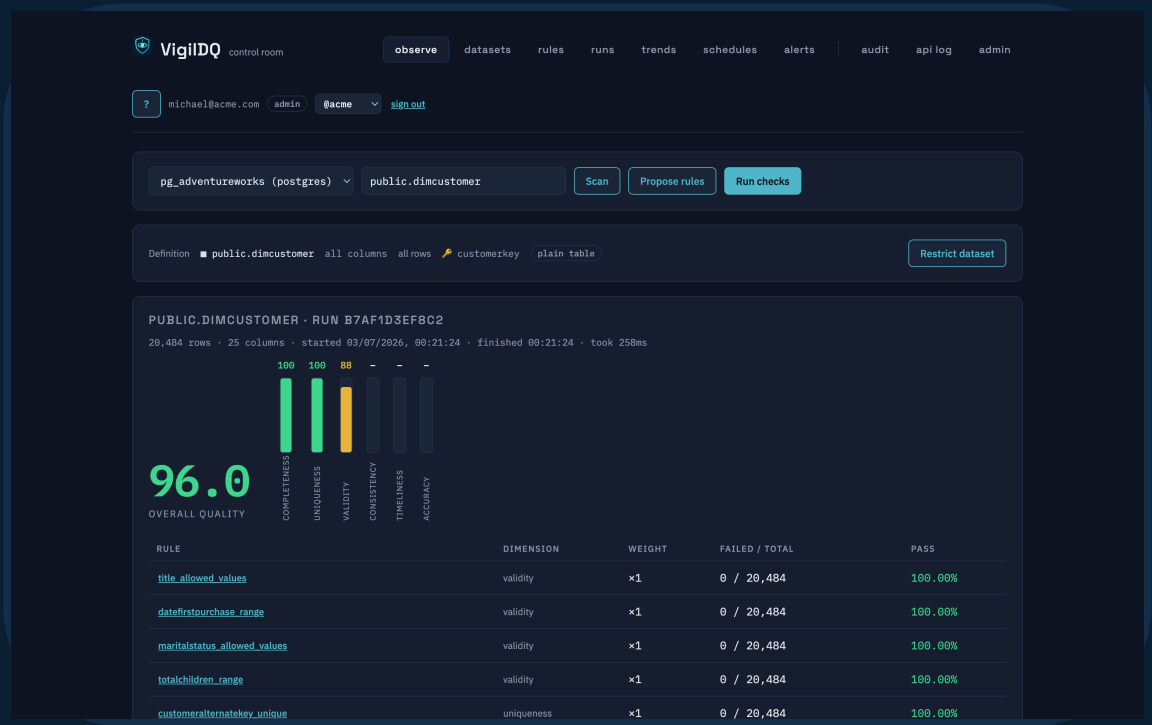
The workflow

Six steps from raw table to governed, continuously-scored dataset — the next ten slides walk through them on the real product.

Connect a warehouse, define the dataset



connections — encrypted at rest, read-only credentials



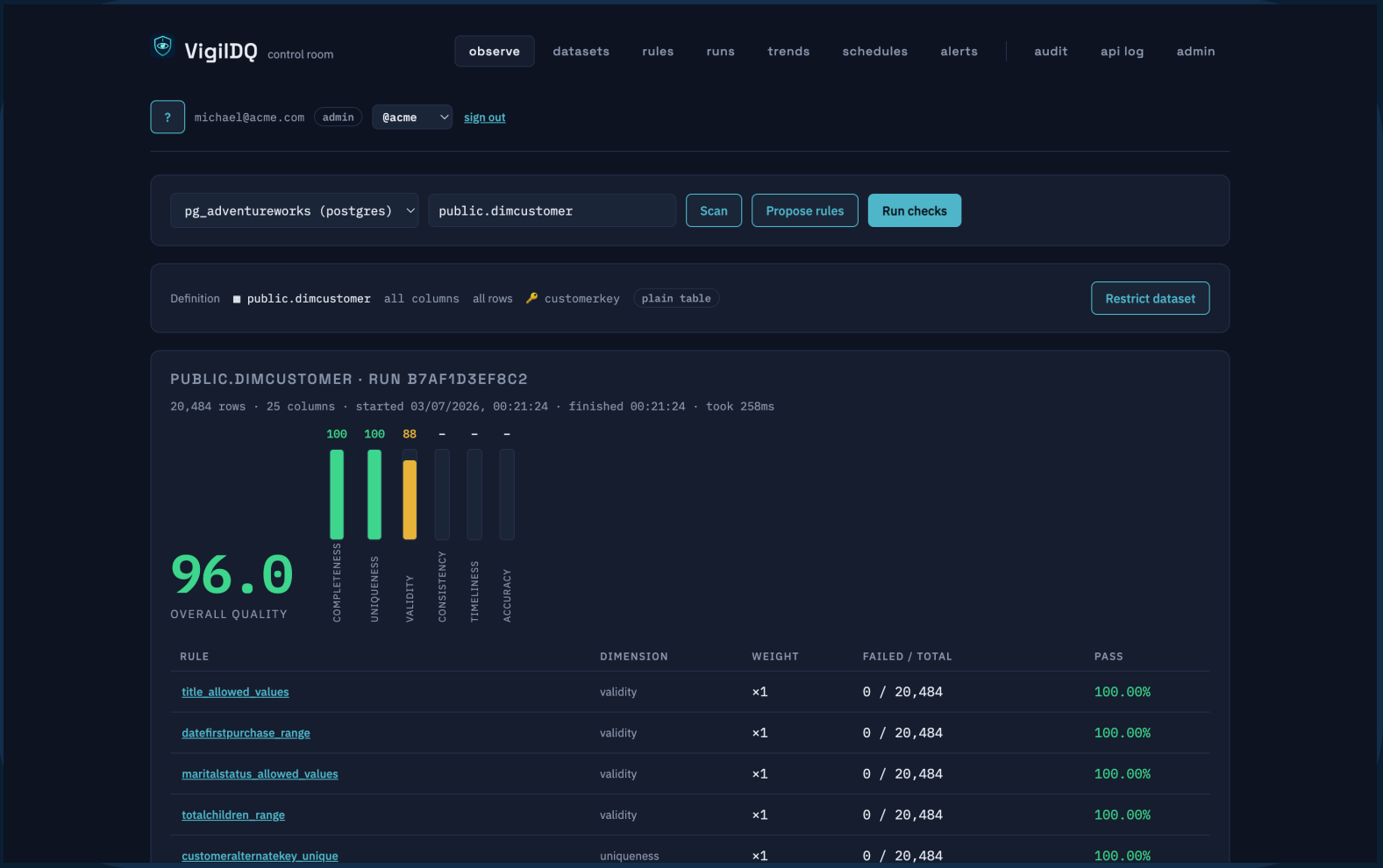
the dataset definition — one click to restrict columns/rows

- Secrets are Fernet-encrypted; rule execution needs read-only grants only.
- Logical datasets: restrict columns and rows (e.g. SCD2 current rows) — rules, profiling and drift all respect the restriction.

WHY IT MATTERS

Your data never leaves the warehouse — VigilDQ pushes queries down and stores results, not rows.

Score the dataset

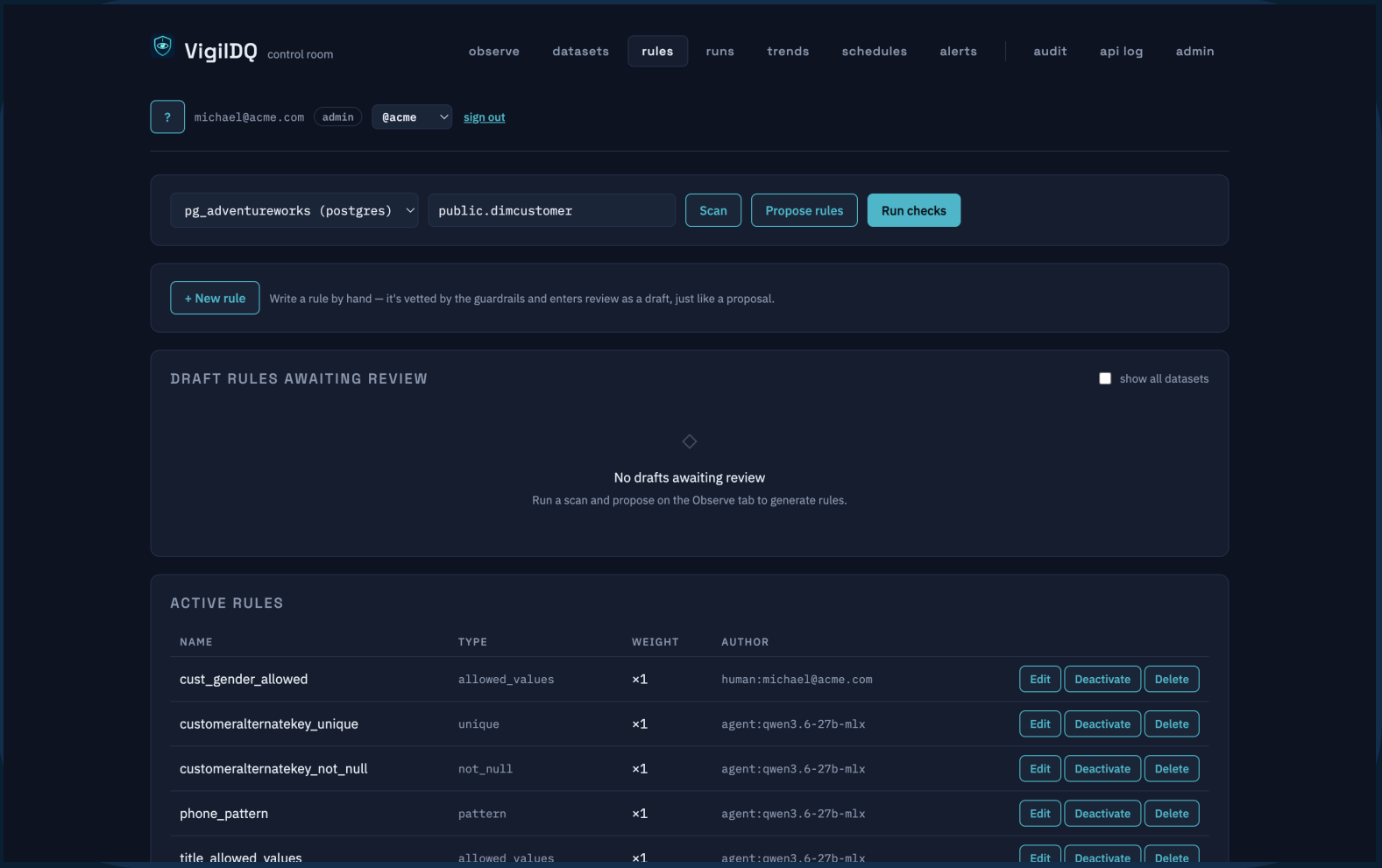


- One 0–100 quality score, decomposed into six dimensions: completeness, uniqueness, validity, consistency, timeliness, accuracy.
- Every rule shows failed/total and its pass ratio — the score is explainable all the way down.
- An erroring rule scores zero and is surfaced — never silently skipped.

WHY IT MATTERS

A number a steering committee can track, with the drill-down an auditor can defend.

Review before anything runs

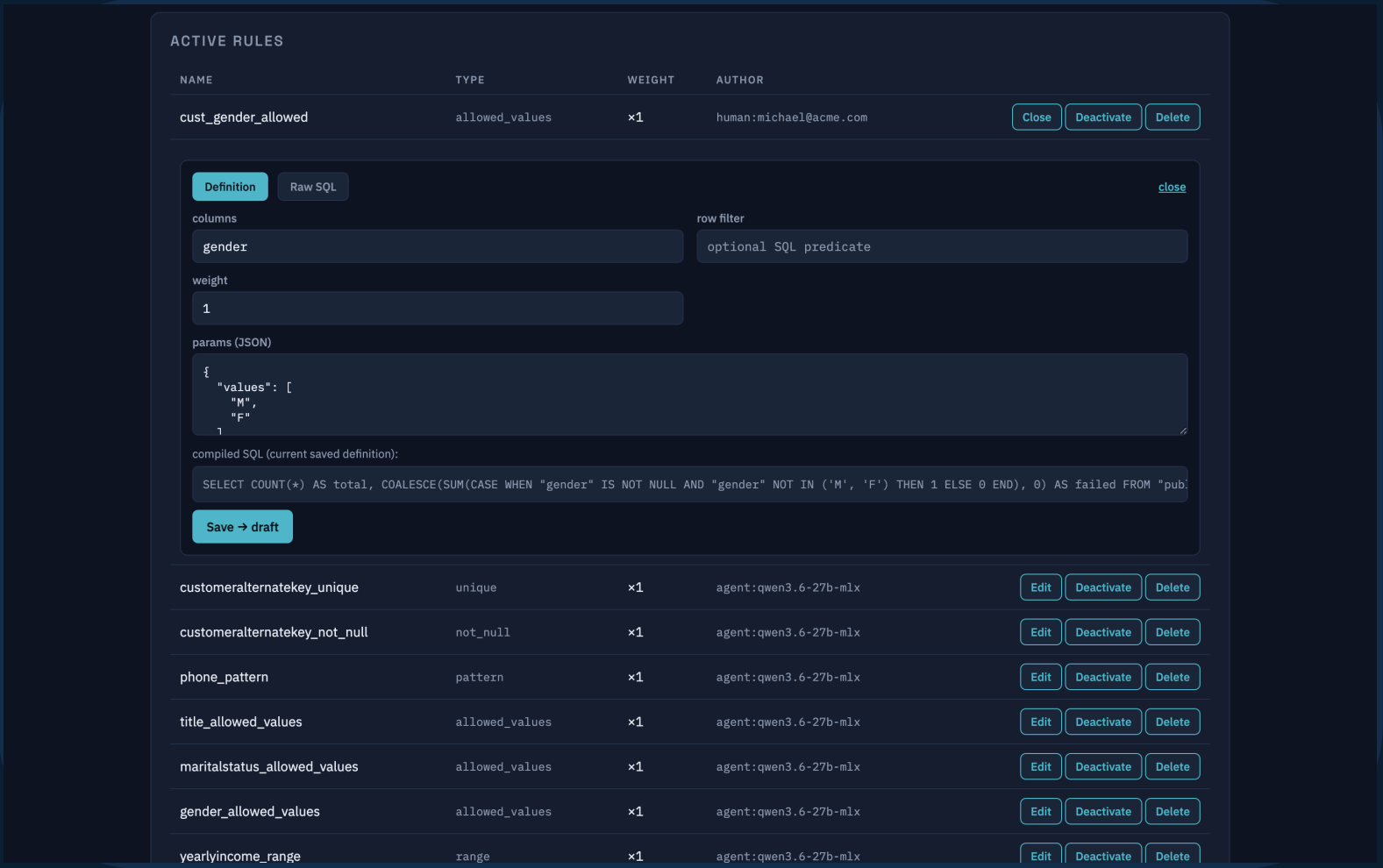


- AI proposals land as drafts with a rationale grounded in the profile.
- Approve, reject, edit or delete — the agent cannot activate anything.
- The approving human is recorded on every rule (human:<email>).
- Hand-written rules go through exactly the same gates.

WHY IT MATTERS

Change control is built in: no AI-authored check ever runs without a named human approving it.

Every rule is inspectable SQL

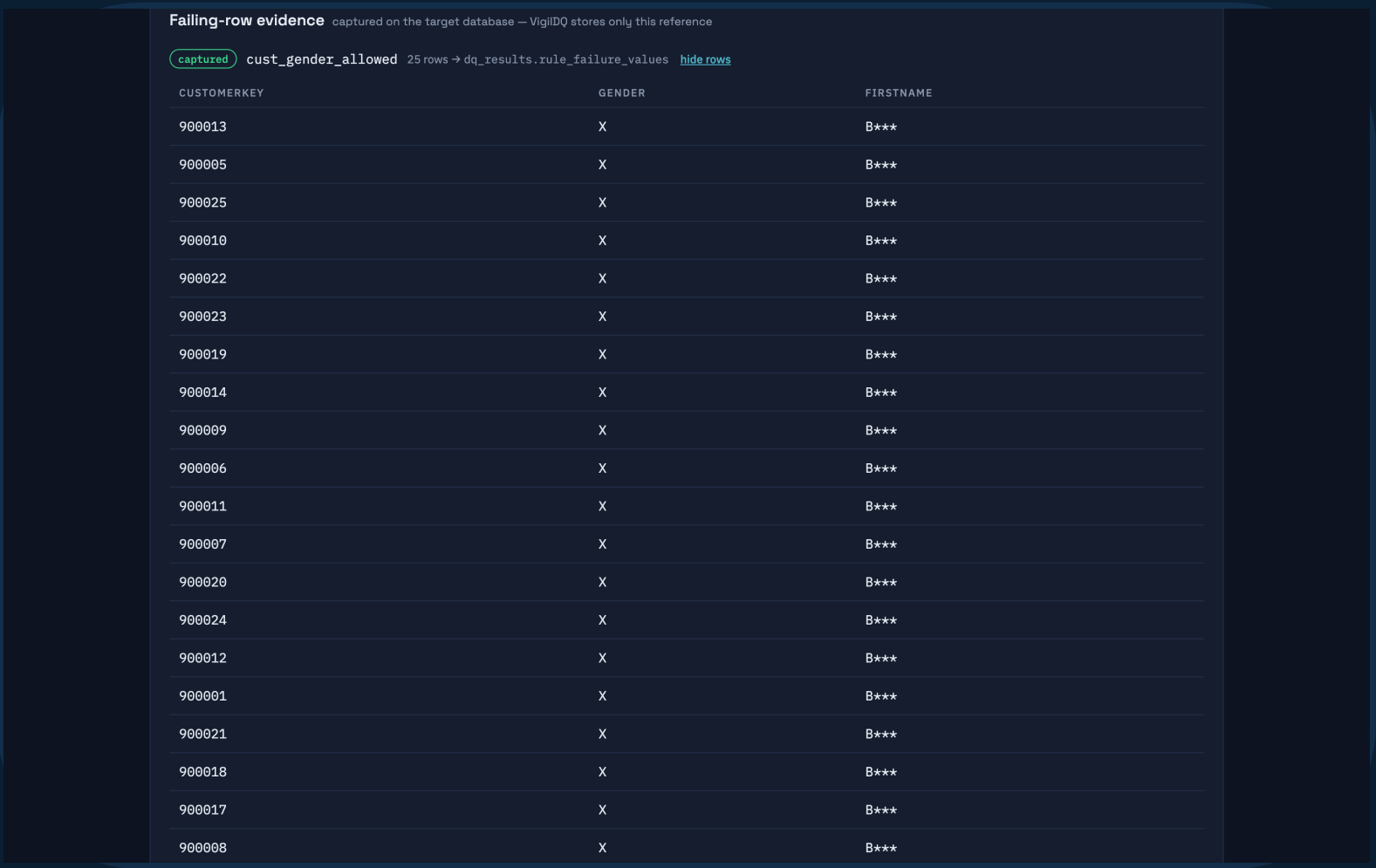


- View the compiled SQL for any rule, per engine dialect.
- Edit the definition — or override with raw SQL for edge cases.
- Any logic change is re-vetted by the guardrails and reset to draft for re-approval.

WHY IT MATTERS

No black boxes: your DBAs can read every control, and edits can't bypass review.

Evidence stays in YOUR database



Failing-row evidence captured on the target database — VigilDQ stores only this reference

captured cust_gender_allowed 25 rows → dq_results.rule_failure_values hide rows

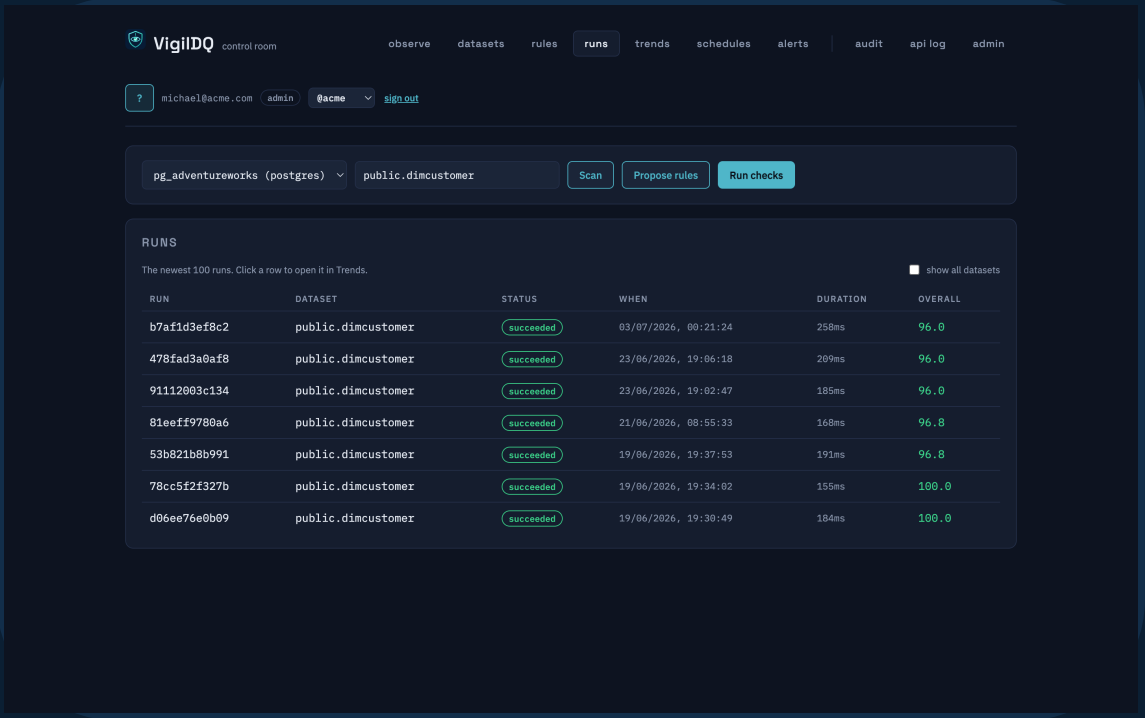
CUSTOMERKEY	GENDER	FIRSTNAME
900013	X	B***
900005	X	B***
900025	X	B***
900010	X	B***
900022	X	B***
900023	X	B***
900019	X	B***
900014	X	B***
900009	X	B***
900006	X	B***
900011	X	B***
900007	X	B***
900020	X	B***
900024	X	B***
900012	X	B***
900001	X	B***
900021	X	B***
900018	X	B***
900017	X	B***
900008	X	B***

- Failing rows are captured into a DQ schema on the target warehouse — keys plus the offending values.
- Capped, fail-soft, and written by a separate writer grant scoped to that schema only.
- The drill-down masks PII at the boundary (names show as initials).

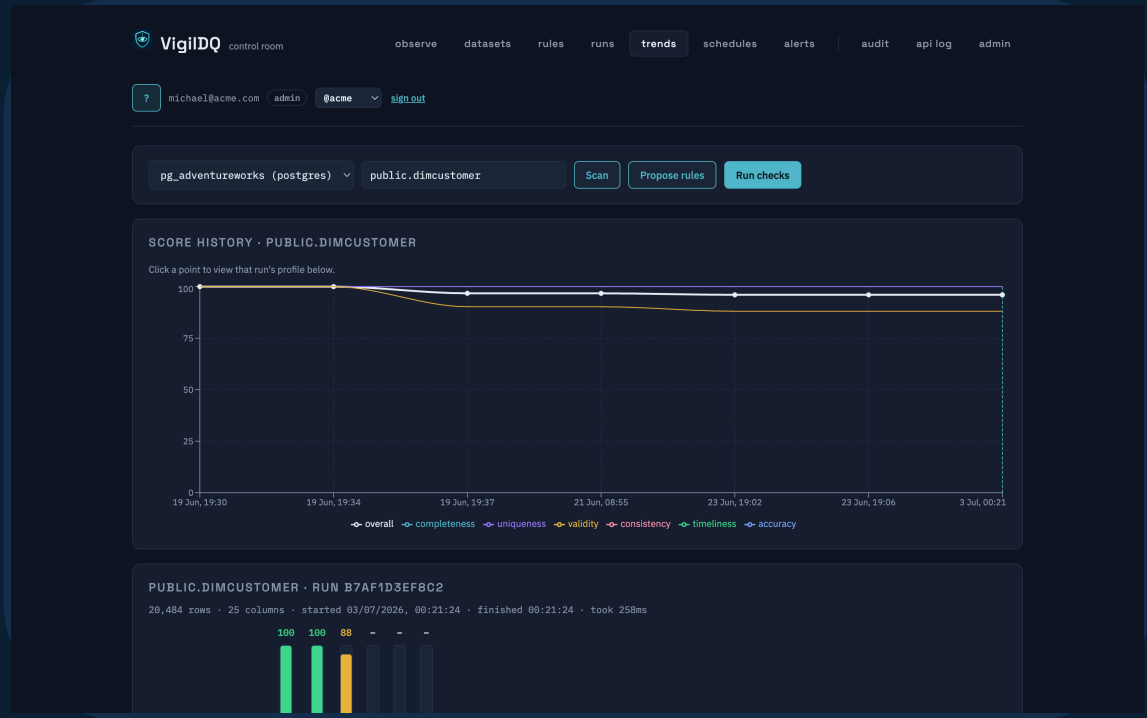
WHY IT MATTERS

When an auditor asks "show me the failing rows", the answer lives in your own database — not in a vendor's cloud.

Track every run, watch the trend



every run: status, duration, score



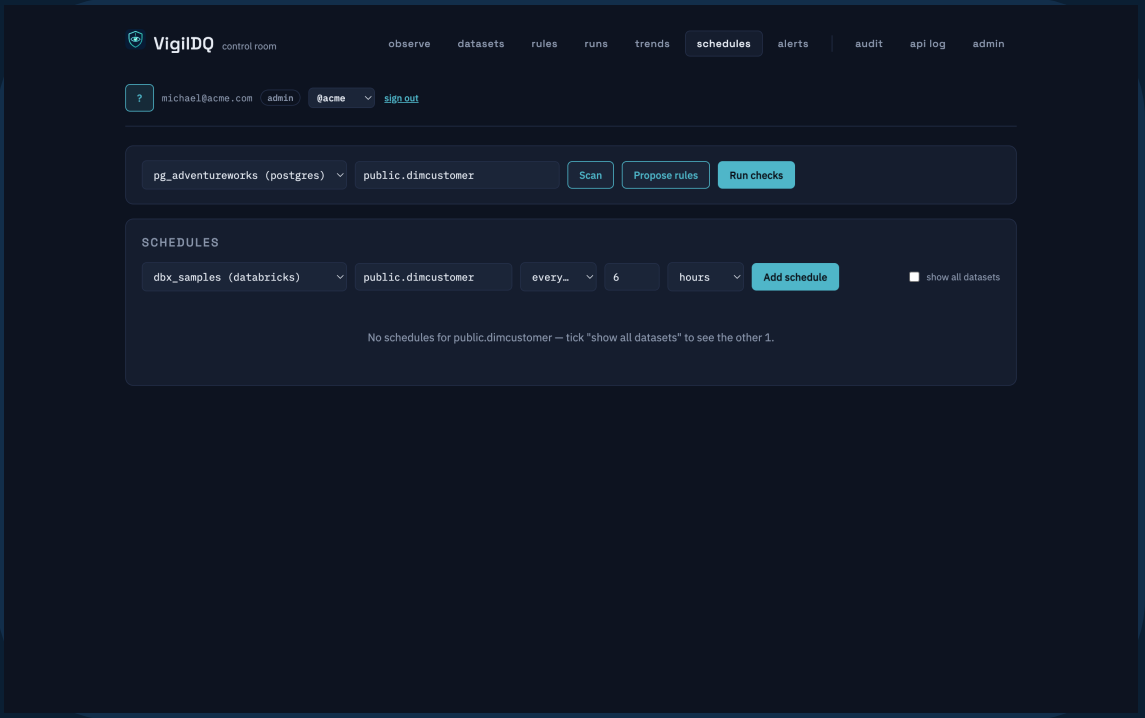
score history + anomaly baselines per dataset

- Full run history with per-run scorecards and captured profiles.
- Drift rules (row count, null ratio, schema) judge against a median baseline — no day-one false alarms.

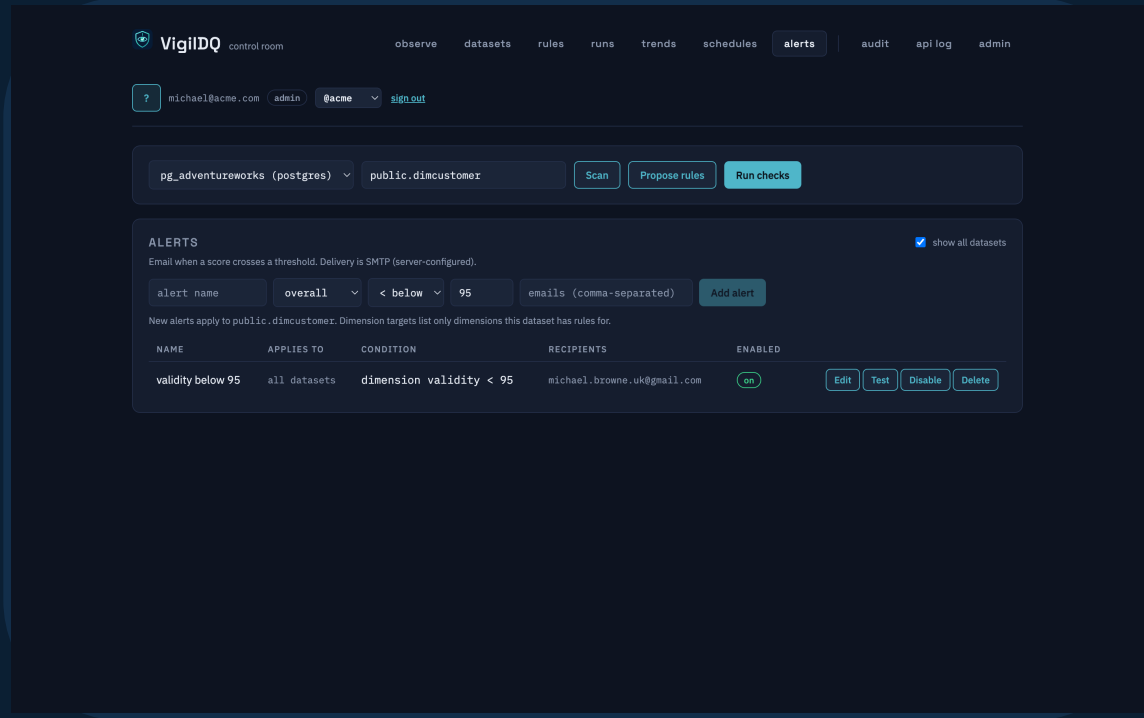
WHY IT MATTERS

Quality becomes a time series — regressions show up as a bend in the line, not a surprise in production.

Automate: schedules, alerts, gates



structured cadences — interval / daily / weekly / monthly



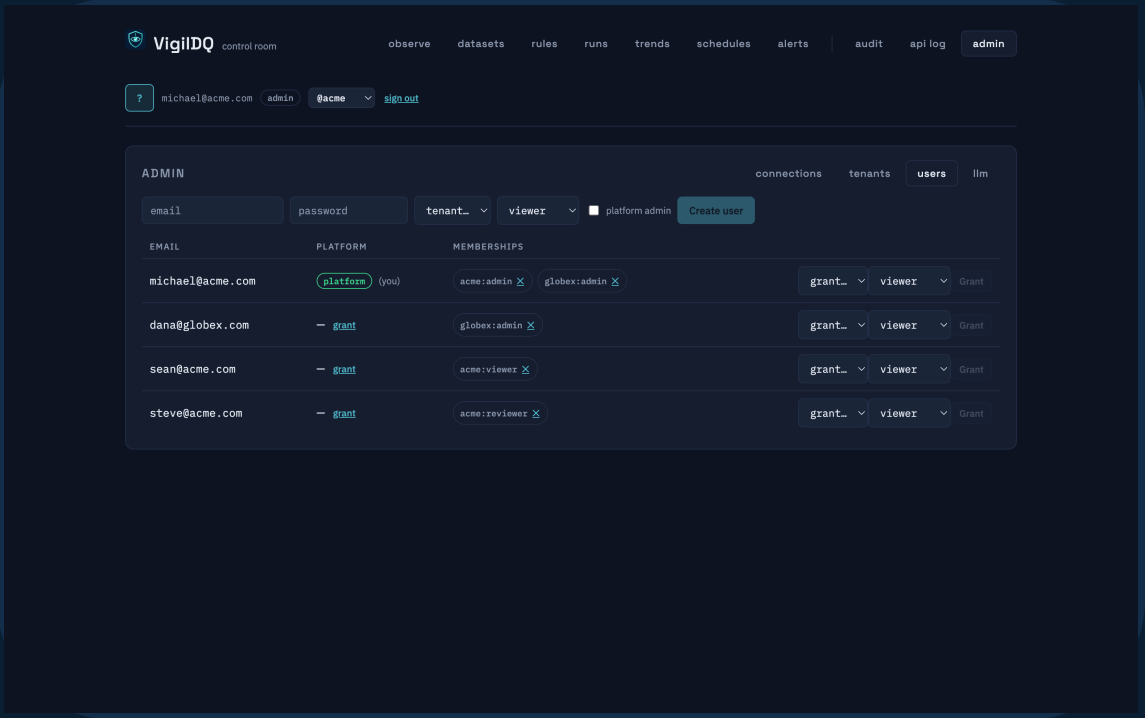
score alerts → email; webhooks for gate-fail / anomaly

- Schedules run each dataset under its own tenant, with recorded health.
- Pipeline quarantine gates return pass/fail verdicts your orchestrator can branch on (Airflow, dbt, ADF...).

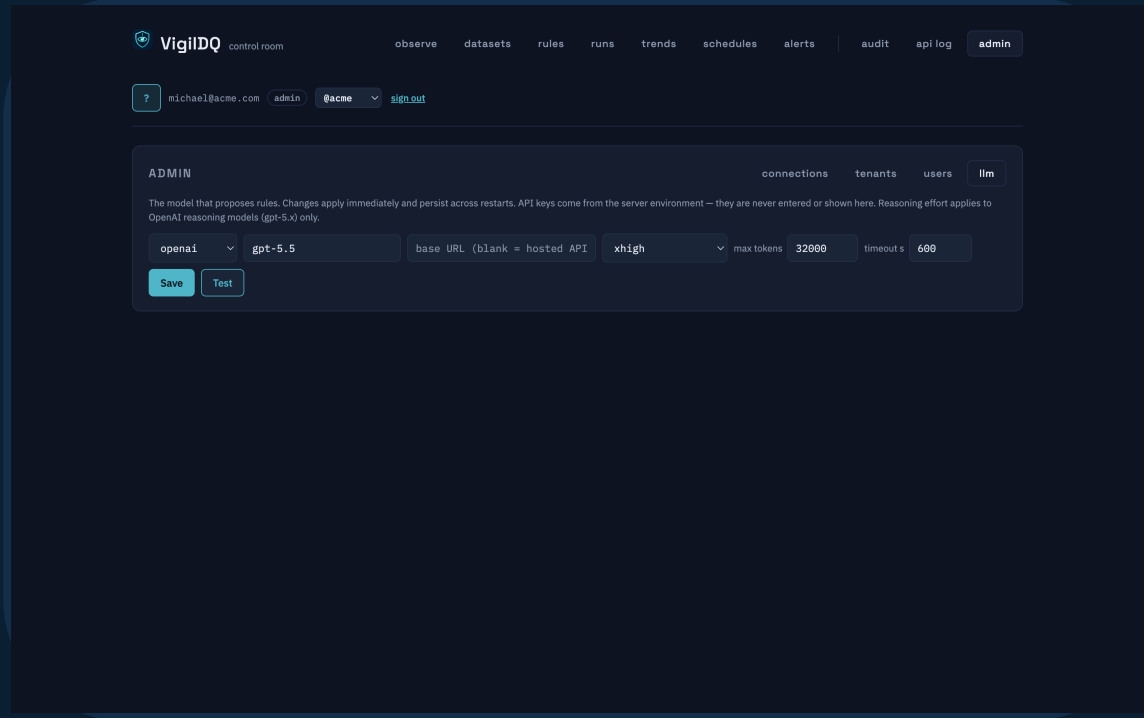
WHY IT MATTERS

Set thresholds once; the platform watches every load and tells you — or stops the pipeline — when quality slips.

Multi-tenant administration



tenants, users, memberships, roles



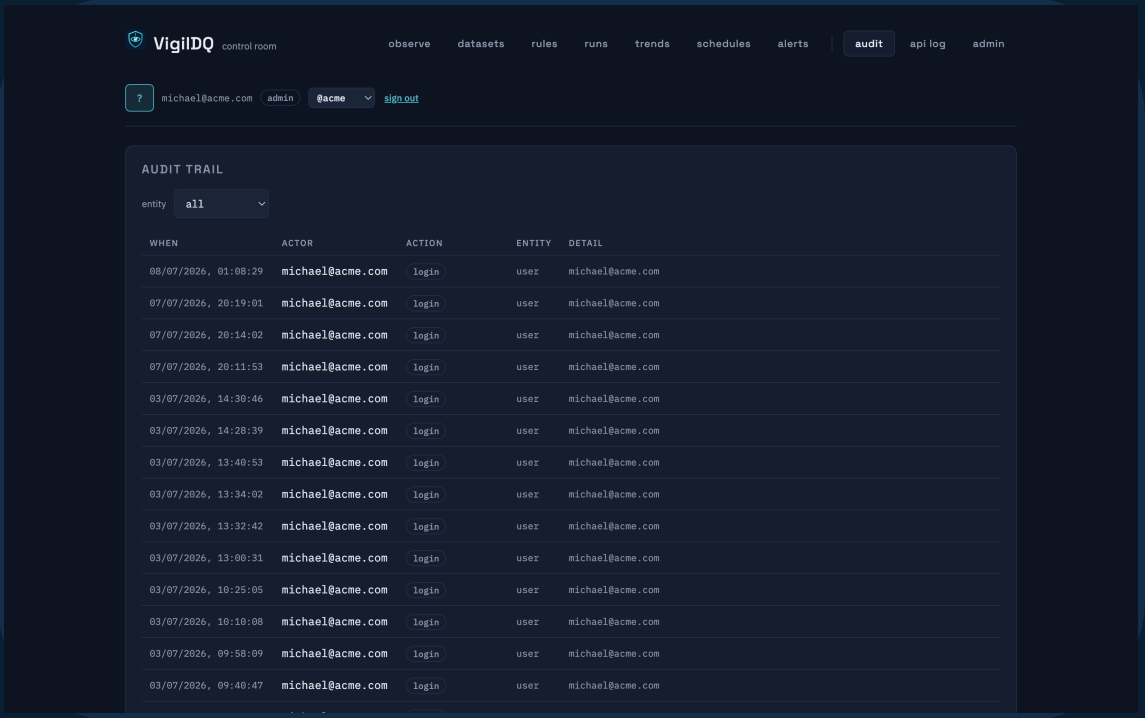
the AI model — switchable at runtime, audited

- Shared-schema multi-tenancy; roles per tenant (admin/reviewer/viewer).
- Choose the LLM in the UI: hosted (Anthropic/OpenAI incl. reasoning models) or fully LOCAL via any OpenAI-compatible server.

WHY IT MATTERS

Bring your own model — including an on-prem LLM, so even the masked profile never leaves your estate.

Audited end to end, scriptable end to end



every mutation: who, what, before/after

the full REST API — everything the UI does

- Approvals, edits, config changes — all appended to an immutable audit trail with before/after snapshots.
- The dashboard is a pure API client: anything you see here, your platform can automate.

WHY IT MATTERS

Compliance gets its evidence; engineering gets an API. Nobody has to trade one for the other.

Designed for regulated data

Guardrails as architecture

- Schema, read-only and dry-run gates — hallucinations cannot pass
- Prompts are advisory; gates make failure modes impossible

Fail closed

- A rule that cannot execute counts as a failed control
- A threshold with no rules is a breach, never a silent pass

Provable masking

- PII masked inside scan() — no unmasked code path exists
- Tests capture the agent's exact prompt and assert zero raw values

Complete audit trail

- Every control names the human who certified it
- The audit log has already been used to reconstruct a metastore

THE NEXT STEP

What a 30-minute demo adds

- The agent proposing rules live — and the review queue filling up.
- A guardrail refusing a bad rule live (a hallucinated column, a sneaky UPDATE).
- A quality gate wired into a pipeline: pass, fail, quarantine.
- All on our sandbox warehouse — nothing to install, no access to your systems.

Watch it catch bad data — live.

Everything in this pack is real and running today. A 30-minute demo adds the parts a PDF can't:
watching the agent propose rules live, a guardrail refusing a bad rule, and a quality gate stopping a bad load.

Arrange a demo → hello@vigildq.com · vigildq.com

The demo runs entirely on our sandbox warehouse — nothing to install, no access to your systems needed.