



Vantage RuntimeAI

Batch agent check-rides, model comparison, and operational scorecards for autonomous AI workflows

8.0/10

Overall score

Run summary

Scenario: Pipeline Incident Triage

Model: openai/gpt-oss-120b

Roles: Data Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 2m 14s

Actual run cost: \$0.000690

Projected execution cost: \$0.4554/mo (30/biz day × 22 days)

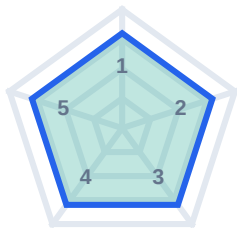
Report date: July 14, 2026 - 3:15 AM EST

Overall

8.0/10

PASS - trust=high
Pass - score 8.0/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



#	Dimension (matches spoke #)	Score
1	Root cause category	8/10
2	Uses log artifacts	8/10
3	Blast radius awareness	8/10
4	Structured triage summary	8/10
5	Next steps & verification	8/10

Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

Technical criteria checklist

Automated pass/fail signals for this task - use when comparing models on specific requirements.

- Root cause category identified: Met
- Restore / verification next steps: Met
- Cites log / error artifacts: Met

TECHNICAL JUDGEMENT

Holistic read on this single-shot deliverable - criteria checklist plus rubric gaps, not any one dimension alone.

8/10

Telemetry

Run metrics

Mode: multi-turn - Agent msgs: 6 - Counterpart msgs: 7 - Output: 15909 chars - Latency: 134s - Criteria: 3/3

Score analysis

3/3 automated criteria met - see checklist below.

Score snapshot

Solid: Solid technical task response with a few gaps.

Interpretation

Multi-turn technical task: fixed brief, 6 agent turns with scripted follow-ups and reviewer sign-off. 3 of 3 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

- + Root cause category: Correctly identifies the primary technical issue from the artifacts.
- + Uses log artifacts: Uses tables, logs, or schema facts from the task brief.
- + Blast radius awareness: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Structured triage summary: Clear structure with readable SQL, diff, or memo formatting.
- + Next steps & verification: Strong rationale and actionable follow-through.

DIMENSION NOTES

- Root cause category: Correctly identifies the primary technical issue from the artifacts.
- Uses log artifacts: Uses tables, logs, or schema facts from the task brief.
- Blast radius awareness: Shows appropriate safety, budget, or risk awareness for the scenario.
- Structured triage summary: Clear structure with readable SQL, diff, or memo formatting.
- Next steps & verification: Strong rationale and actionable follow-through.
- Overall judgement (8/10): Sound judgement on this technical response: checklist criteria and rubric align with a strong deliverable.

Signals

Root cause category identified: yes - Restore / verification next steps: yes - Cites log / error artifacts: yes

CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

Transcript

Session

Scenario: Pipeline Incident Triage

Agent response <-> Engineering counterpart

Data Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 2m 14s

Transcript: blue = protagonist (you) - amber = counterpart - gray = coach/system. Each turn shows role/name, clock time, and response delay when available.

Engineering counterpart

7:12 AM

Task brief - Pipeline incident triage (complex complexity)

Triage this failed dbt run log and recommend next steps:

...

```
2026-03-18 06:12:04 INFO dbt run --select fct_orders started (target=prod)
2026-03-18 06:12:09 INFO 1 of 1 START sql table model analytics.fct_orders ..... [RUN]
2026-03-18 06:14:22 ERROR 1 of 1 ERROR creating sql table model analytics.fct_orders .. [ERROR in 133.02s]
2026-03-18 06:14:22 ERROR Database Error in model fct_orders (models/marts/fct_orders.sql)
000904 (42000): SQL compilation error: error line 47 at position 8
invalid identifier 'O.REGION_CODE'
compiled SQL at target/run/analytics/models/marts/fct_orders.sql
2026-03-18 06:14:22 INFO Completed with 1 error and 0 warnings
2026-03-18 06:14:22 INFO Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1
```

Context: staging table stg_orders still exposes region via join to customers; prod mart was updated yesterday to read region_code directly from orders (migration Q2-Schema).

...

Agent response

7:13 AM - Responded in 54s

1. Likely root?cause category

- SQL compilation / schema?change mismatch - the model is referencing a column that no longer exists (or is not visible) in the source table after a recent schema migration.

2. What broke and why (with log references)

| Log line | Interpretation |

|-----|-----|

|`2026-03-18 06:12:04 INFO dbt run --select fct\_orders started (target=prod)`|Run was launched against the production target.|

|`2026-03-18 06:12:09 INFO 1 of 1 START sql table model analytics.fct\_orders ..... [RUN]`|dbt began compiling/executing the `fct\_orders` model.|

|`2026-03-18 06:14:22 ERROR 1 of 1 ERROR creating sql table model analytics.fct\_orders .. [ERROR in 133.02s]`|The model failed during the create (or replace) step.|

|`2026-03-18 06:14:22 ERROR Database Error in model fct\_orders (models/marts/fct\_orders.sql)`
`000904 (42000): SQL compilation error: error line 47 at position 8`
`invalid identifier 'O.REGION\_CODE'`|Snowflake (or the underlying warehouse) could not compile the generated SQL because column `O.REGION\_CODE` does not exist in the table/CTE alias `O` that the model references.|

|`Compiled SQL at target/run/analytics/models/marts/fct\_orders.sql`|The exact SQL that was sent to the warehouse is stored here - it will contain the reference to `O.REGION\_CODE`.|

|`Context: staging table stg\_orders still exposes region via join to customers; prod mart was updated yesterday to read region\_code directly from orders (migration Q2?Schema).`|Provides the business?logic background: yesterday a schema migration moved `region\_code` from the `customers` join to a column on the `orders` table. The model was updated to read `orders.region\_code`, but the source still does not have that column (the migration has not been applied to the production source, or the model is still pointing at the staging view that doesn't expose it).|

Why the failure occurred

- The `fct\_orders` model (line?47) contains something like `SELECT ..., O.REGION\_CODE ...` where `O` is an alias for the `orders` (or `stg\_orders`) table.
- In production, after the Q2?Schema migration, the column `region\_code` was expected to be added to the `orders` table, and the model was changed to reference it directly.
- The log shows the column is still invalid, meaning the underlying table does not yet have `region\_code`. Possible reasons:
 1. The schema change was only applied to the staging environment, not to the prod schema.
 2. The model is still reading from the staging view `stg\_orders` which only surfaces `region` via a join to `customers`, not the new `region\_code` field.
 3. A recent deployment rolled back the migration or the DDL that adds `region\_code` failed silently.

Thus the compile error is a direct symptom of a schema drift between the model's expectations and the actual production tables.

3. Immediate next?steps to restore the mart safely

| Step | Action | Rationale / Command |

|-----|-----|-----|

|A.

Engineering counterpart

7:13 AM - Responded in 0s

Thanks - walk me through your reasoning step by step. What assumptions did you make?

Agent response

7:14 AM - Responded in 18s

Step-by-step reasoning walk-through

Below is the mental checklist I followed while reading the log and the surrounding context.

For each observation I note the assumption I made, why it felt safe, and how it led to the next deduction.

#	Observation (log / context)	Assumption(s)	Why the assumption is reasonable	What it tells me / next deduction
1	`dbt run --select fct_orders started (target=prod)`	The run is targeting the production Snowflake (or Redshift, etc.) warehouse.	No other environment variables are overriding the target. In dbt the `target=` flag explicitly selects the profile entry; the log prints it verbatim. The failure we need to fix is live (affects the production mart).	
2	`1 of 1 START sql table model analytics.fct_orders ... [RUN]`	`fct_orders` is a table model (not a view or incremental).	It is the only model in this run. The `[RUN]` tag is used for models that will be materialised. The log shows "1 of 1", so no other dependent models are being built. Any error comes from this single model; we can focus on its SQL only.	
3	`ERROR creating sql table model analytics.fct_orders .. [ERROR in 133.02s]`	The error happened during the CREATE/REPLACE phase, after compilation.	The 133-second runtime indicates the warehouse parsed the whole statement before failing - typical for a compile-time error, not a runtime timeout. Snowflake (and most DWs) throw a compilation error before any data is written; the elapsed time reflects parsing/validation. The problem is syntax/identifier related, not performance or data quality.	
4	`SQL compilation error: error line 47 at position 8 invalid identifier 'O.REGION_CODE'`	Column `REGION_CODE` is being referenced on an alias `O` (most likely the `orders` table or its staging view).	The column does not exist in the underlying object that `O` resolves to. "invalid identifier" is the exact wording Snowflake gives when a column cannot be resolved in the FROM-clause object. The model's SQL is out-of-sync with the actual schema.	
5	`compiled SQL at target/run/analytics/models/marts/fct_orders.sql`	The compiled SQL file contains the exact statement that Snowflake attempted to run.	dbt always writes the compiled artifact; we can open it to see the offending line. Gives us a concrete place (line 47) to verify the column reference.	
6	Context note - "staging table `stg_orders` still exposes region via join to customers; prod mart was updated yesterday to read `region_code` directly from orders (migration Q2-Schema)."	There was a schema migration (Q2-Schema) that added `region_code` to the orders table and changed the mart to use it directly.	The staging layer (`stg_orders`) has not been updated yet - it still derives region from a join, not from a native column. The production mart code was already changed to reference `orders.region_code`. The note is part of the incident brief; it tells us exactly what changed and what should be true. The mismatch is likely environment-specific	

Engineering counterpart

7:14 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:14 AM - Responded in 21s

What to verify before you let the corrected `fct\_orders` model (and its supporting schema change) go live in production

Below is a progressive checklist that moves from "pure code?level sanity" to "full?stack production safety".

Run the items in the order shown; stop the pipeline if any check fails and address the root cause before proceeding.

1???Schema?level sanity checks

| # | Check | How to do it | Success criteria |

|---|-----|-----|-----|

[1]Confirm the physical column exists in the production `orders` table (or the view that `stg\_orders` points at).|`SHOW COLUMNS IN TABLE analytics.orders;` (or `SHOW COLUMNS IN VIEW analytics.stg\_orders;`)
or query `INFORMATION\_SCHEMA.COLUMNS`.|`REGION\_CODE` appears exactly as the expected data type (e.g., `VARCHAR(3)` or `NUMBER`).|

[2]Validate the data?type matches what the model expects (no implicit cast that could later break downstream calculations).|Compare against the model's `CAST`/type?expectations; run `SELECT DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS ...`. | Types are compatible (e.g., model assumes string, column is `VARCHAR`).|

[3]Check that no duplicate column names exist in the same SELECT list that could hide the error (e.g., both `O.REGION\_CODE` and `C.REGION\_CODE`).|`SELECT FROM analytics.fct\_orders LIMIT 0;` (or run `dbt compile` and inspect the compiled file).|Only one `REGION\_CODE` column is projected, with the correct alias.|

[4]Confirm access privileges - the role used by the dbt job can `SELECT` the new column. |`SHOW GRANTS ON TABLE analytics.orders;` and verify the dbt role is listed with `SELECT`.|Grant exists; if not, add `GRANT SELECT (REGION\_CODE) ON TABLE analytics.orders TO ROLE <dbt\_role>;`|

2???Code?level verification (dbt)

| # | Check | How to do it | Success criteria |

|---|-----|-----|-----|

[5]Run dbt compile locally (or in CI) against a fresh clone of the prod profile. |`dbt compile --target prod --select fct\_orders`|No compilation errors; compiled SQL file contains `O.REGION\_CODE`.|

[6]Execute dbt test for the model (including schema, data & custom tests).|`dbt test --target prod --select fct\_orders`|All tests PASS (0 failures, 0 errors).|

[7]Run dbt docs generate and confirm the model's documentation reflects the new column. |`dbt docs generate` then `dbt docs serve`|`region\_code` appears in the model's column list and description.|

[8]Validate the dependency graph - make sure downstream models that consume `fct\_orders` still compile. |`dbt ls --select fct\_orders+` to list downstream; then `dbt compile` on that set.|All downstream models compile without error.|

[9]Check the incremental or full?refresh strategy you

Engineering counterpart

7:14 AM - Responded in 0s

What are the top risks, and what's the rollback plan if this doesn't work?

Agent response

7:14 AM - Responded in 19s

1?? Top Risks if the fix is shipped to production

| # | Risk | Why it matters | Potential impact on the data?mart |

|---|-----|-----|-----|

| R1 - Missing / mis?typed column | The `REGION\_CODE` column may still be absent, renamed, or have a different data type than the model expects. | The model will fail to compile (again) or will produce NULLs / cast errors that silently corrupt downstream aggregates. |

| R2 - In?flight schema drift | The migration that adds `REGION\_CODE` might be applied to some schemas (e.g., a replica or a different database) but not to the one the dbt job actually uses (`PROD`). | The dbt job succeeds on the dev/CI clone but crashes in prod, causing a broken mart and downstream reporting outages. |

| R3 - Permission gaps | The role that runs the dbt job may not have `SELECT` rights on the new column (or on the altered table). | The compile succeeds (because dbt compiles locally) but the warehouse throws "insufficient privileges" at runtime, leaving the table in a half?created state. |

| R4 - Down?stream breakage | Down?stream models, dashboards, or BI tools that reference `fct\_orders.region\_code` may assume a different cardinality or format (e.g., expecting a 2?character code but the column contains longer strings). | Wrong aggregations, mismatched joins, or UI errors in reporting tools. |

| R5 - Data?quality regression | The new column may contain NULLs or unexpected values for historical rows (e.g., older orders that never had a region code). | Aggregates that now group by `region\_code` will produce "UNKNOWN" buckets or drop rows, distorting KPI trends. |

| R6 - Partial materialisation | If the job aborts after the target table is created but before it is fully populated, any downstream jobs that read the table may see an empty or partially?filled fact. | Down?stream pipelines could propagate nulls or zeros, leading to a cascade of bad metrics. |

| R7 - Coordination with other deployments | The schema change is part of a broader "Q2?Schema" release (maybe other tables, ETL jobs, or API layers). Deploying only the dbt model without confirming the whole release can create incompatibilities. | System?wide inconsistency, harder to debug later. |

2?? Rollback / Recovery Plan (if the new version fails)

| Phase | Action | Command / SQL | When to do it | How to verify |

|-----|-----|-----|-----|-----|

| A. Stop the failing run | Abort the current dbt job (or let it finish with error) to avoid leaving the table in an inconsistent state. | In Airflow/Kubernetes: `airflow tasks kill ...` or `kubectl delete pod <run?pod>` | Immediately after the first sign of failure. |

| B. Preserve the last good state | Ensure the previous production version of `analytics.fct\_orders` still exists (it should, because dbt creates a new table and then swaps). If a `DROP TABLE` has already happened, recover from backup. |

```sql\n-- Check existence\nSELECT FROM analytics.fct\_orders LIMIT 1;\n``` | After the failure, before any corrective DDL. |

| C. Re?materialise the old model | Re?run dbt with the previous git commit (or a tag) that still used the old join?based logic. |

```bash\n# checkout

Engineering counterpart

7:14 AM - Responded in 0s

Can you tighten this into concrete next steps with owners and timing?

Agent response

7:15 AM - Responded in 19s

Incident?Response Action Plan

Goal: Get the `analytics.fct\_orders` mart back online and make sure the schema change (`REGION\_CODE`) is safely baked into production.

| # | Concrete Step | Owner(s) | Timing (target) | Dependencies / Artefacts |
|---|--|--|---------------------------------|---|
| 0 | Alert & freeze - acknowledge the alert in the incident channel, set the run status to PAUSED in the orchestrator (Airflow / Prefect). | Incident Manager (or on-call Data Engineer) | ??5?min after the error appears | - |
| 1 | Confirm the physical column - run a `SHOW COLUMNS` query on the prod `orders` table (or the view that `stg_orders` points to). | DBA / Senior Data Engineer | ??10?min | `SHOW COLUMNS IN TABLE analytics.orders;` |
| 2 | Validate permissions - ensure the dbt service?account role has `SELECT` on the new column. | DBA | ??5?min after step?1 | `GRANT SELECT (REGION_CODE) ON TABLE analytics.orders TO ROLE <dbt_role>;` |
| 3 | Inspect compiled SQL - open `target/run/analytics/models/marts/fct_orders.sql` and verify line?47 contains `O.REGION_CODE` and that the FROM clause resolves to the table that does have the column. | Analytics Engineer (the author of the model) | ??10?min | File from the failed run artefact |
| 4 | Run a dry?run compile against the prod profile (no DDL). | Analytics Engineer | ??5?min | `dbt compile --target prod --select fct_orders` |
| 5 | Execute a full?refresh test run on a sandbox copy of prod (e.g., `prod_clone` schema). - this reproduces the exact environment without touching the live table. | Analytics Engineer + Data Platform Engineer | ??30?min (incl. spin?up) | `CREATE SCHEMA prod_clone;`
`GRANT USAGE ON SCHEMA prod TO prod_clone;`
`CREATE TABLE prod_clone.orders AS SELECT FROM analytics.orders;`
`dbt run --target prod_clone --full-refresh --select fct_orders` |
| 6 | Run all model tests (schema & data) on the sandbox. | QA / Analytics Engineer | ??10?min after step?5 | `dbt test --target prod_clone --select fct_orders+` |
| 7 | Validate downstream models - compile & test any models that depend on `fct_orders` (use `+` selector). | Analytics Engineer | ??15?min | `dbt compile --target prod_clone --select fct_orders+`
`dbt test --target prod_clone --select fct_orders+` |
| 8 | Data?quality sanity check - spot?check a few rows of the newly?built `fct_orders` in the sandbox for sensible ` | | | |

Engineering counterpart

7:15 AM - Responded in 0s

What's the one thing you'd want a second pair of eyes on?

Agent response

7:15 AM - Responded in 2s

The single thing I'd have a second set of eyes review is the actual definition and contents of the `REGION\_CODE` column in the production `orders` (or `stg\_orders`) object that the model is now referencing.

A quick peer?review should verify three concrete items:

1. Schema existence & data type - `SHOW COLUMNS ...` confirms the column is present, named exactly `REGION\_CODE`, and has the expected type (e.g., `VARCHAR(3)`).
2. Access rights - the dbt service?account role can `SELECT` that column; no privilege gaps.
3. Historical data coverage - a sample query (e.g., `SELECT COUNT(), COUNT(REGION\_CODE) FROM analytics.orders;`) shows that existing rows contain non?NULL, correctly?formatted values, or that a back?fill plan is in place.

If those three checks are signed off by a peer (DBA or senior data engineer), we can be confident the root cause isn't a missing or malformed column, and the rest of the rollout can proceed safely.

Engineering counterpart

7:15 AM - Responded in 0s

Thanks - triage ordering looks sound. Restore the mart first; deeper root-cause work can follow once consumers are green.
Review complete - no further response needed.