



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-4.1-mini

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

Duration: 48s

Actual run cost: \$0.006960

Projected execution cost: \$4.59/mo (30/biz day × 22 days)

Report date: July 14, 2026 - 3:12 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: 12991 chars - Latency: 48s - 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: 48s

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

Engineering counterpart

7:11 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

7:11 AM - Responded in 8s

(1) Likely root cause category:

Schema/Model mismatch due to recent data model update not reflected in downstream transformations.

(2) What broke and why:

The model `analytics.fct\_orders` failed with a SQL compilation error:

...

```
invalid identifier 'O.REGION_CODE'
```

...

(Log line: `2026-03-18 06:14:22 ERROR Database Error in model fct\_orders ... invalid identifier 'O.REGION\_CODE'`)

This indicates that the compiled SQL references a column `region\_code` prefixed as `O.REGION\_CODE` (alias `O` likely stands for the orders table), but that column is not present in the source table or staging table at runtime.

The context note states that the prod mart was updated yesterday to read `region\_code` directly from `orders` as part of a Q2-schema migration, but the staging table `stg\_orders` still exposes `region\_code` via a join to `customers`. This discrepancy suggests that the new column `region\_code` was not yet added to or made available in the `orders` source table or the staging model, causing the compiled SQL referencing `O.REGION\_CODE` to fail compilation.

(3) Immediate next steps to restore the mart safely:

- Verify if the `region\_code` column physically exists in the `orders` source table in the production database (the table aliased as `O` in the model SQL). If not, the migration to add this column has not been fully applied or synchronized.
- If `region\_code` is missing in the `orders` source, revert the model `fct\_orders` to the previous version that joins to `customers` for `region\_code` until the source schema update completes successfully.
- Alternatively, update or rebuild the staging table `stg\_orders` to expose `region\_code` directly from `orders` if the column exists but staging is outdated.
- Run a full dbt run to confirm the fix before re-enabling the pipeline.
- Coordinate with the schema migration team to confirm the rollout status of the `region\_code` column on the orders table to prevent such downstream failures.

Engineering counterpart
7:11 AM - Responded in 0s

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

Agent response

7:12 AM - Responded in 12s

Certainly! Here's my reasoning step-by-step along with the assumptions I made:

1. Identifying the error:

The logs show a failed dbt run of a single model `analytics.fct\_orders` with a compilation error:

```
invalid identifier 'O.REGION_CODE'
```

This indicates the SQL references a column `region\_code` on the table aliased as `O` (likely the orders table), but the database cannot find that column during compilation.

2. Interpreting the context note:

The user-provided context states:

> "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)."

This implies:

- Previously, `region\_code` was derived by joining `orders` with `customers` (a foreign key join to get region info).
- The prod mart SQL was updated recently to read `region\_code` directly from the `orders` table (presumably a new column added as part of a Q2 schema migration).
- However, the staging table `stg\_orders` has not been updated to reflect this change and still exposes `region\_code` via the old join.

3. Identifying the root cause:

The failure is caused by referencing `O.REGION\_CODE` in the SQL while the actual underlying table or staging model does not have that column yet. This causes a compilation error in Snowflake (or similar SQL engine).

Therefore, the issue is a mismatch between the data model (dbt model) and the physical schema - the model expects `region\_code` in `orders`, but the column is not there yet.

4. Assumptions I made:

- The alias `O` in SQL stands for the `orders` source table or staging table (common alias in marts), so referencing `O.REGION\_CODE` means `region\_code` must exist on that table.
- The `region\_code` column has not been fully deployed or is not yet available in the `orders` source table or staging table, causing the compilation error.
- The staging table `stg\_orders` is upstream from `fct\_orders` and is currently lagging behind the schema migration.
- The dbt run failed during compilation, which implies this is not a runtime data error but a schema/reference error in SQL code.

5. Conclusion and recommended immediate actions:

- Since the model expects a schema change that isn't live yet, reverting to the old model logic or updating upstream staging tables to match the new schema will resolve the issue.
- Confirming the status of the `region\_code` column rollout is critical to prevent repeated failures.
- A careful controlled rollback or fix is safer than rushing further changes.

Summary:

I connected the error message (`invalid identifier 'O.REGION\_CODE'`) to the information that the prod mart was recently updated to reference this column directly on `orders`. Given `stg\_orders` still exposes `region\_code` via join, the physical schema and transformation logic are out of sync. This mismatch leads to the compilation error. Thus, the root cause is a partial/incomplete data model migration, and the immediate fix is to either revert or align staging/source tables with the new schema.

Engineering counterpart

7:12 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:12 AM - Responded in 8s

Before shipping any fix or updated model to production, I would verify the following to ensure stability and correctness:

1. Physical schema validation:

- Confirm the `region\_code` column exists in the `orders` source table in the production database (check with `DESCRIBE TABLE orders` or equivalent).
- Validate the column has the correct data type and is populated as expected (no null critical values).

2. Staging model alignment:

- Check that `stg\_orders` (the staging table/model) includes `region\_code` sourced directly from `orders` if the new model expects it there.
- If `stg\_orders` still joins to `customers` for `region\_code`, either update the staging model or revert the mart model to the old logic.

3. Code correctness and compilation:

- Run a full local or CI dbt compile and run in a dev or staging environment against a production-like dataset to ensure no compilation or runtime errors.
- Check the compiled SQL (at `target/run/...`) for the `fct\_orders` model to confirm it references `region\_code` correctly.

4. Data correctness and regression testing:

- Run data quality checks comparing the previous and new outputs, especially on the `region\_code` column and related metrics.
- Verify that joined data (region info) matches expectations and no unexpected nulls or mismatches appear.

5. Deployment checklist:

- Confirm all dependent models or reports downstream are compatible with the change or updated accordingly.
- Communicate schema changes and deployment plans with stakeholders (analytics, BI, etc.).
- Have a rollback plan ready if issues occur post-deployment.

6. Pipeline health check:

- Run the full dbt run command against production after staging validation to confirm the entire pipeline runs cleanly without warnings or errors.

In summary:

I would verify the underlying schema changes have been fully deployed and are consistent with the model code, confirm staging models and transformations align with the new schema, and test thoroughly in a safe environment before shipping the fix to production. This approach minimizes impact and ensures safe restoration of the mart.

Engineering counterpart

7:12 AM - Responded in 0s

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

Agent response

7:12 AM - Responded in 8s

Top risks:

1. Source schema incompleteness or delay:

The `region\_code` column may not be fully deployed or backfilled in the `orders` source table, causing silent data quality issues or failures.

2. Staging model/schema mismatch:

Staging table `stg\_orders` may not be updated or refreshed properly to expose `region\_code` directly, causing downstream models to fail or produce incorrect data.

3. Downstream dependency failures:

Other marts, reports, or dashboards relying on `fct\_orders` or `region\_code` may break or show inconsistent data due to schema or semantic changes.

4. Data correctness issues:

New `region\_code` data from `orders` could differ from the previous join-based values, leading to analytic regressions or business confusion.

5. Insufficient testing coverage:

Rushing deployment without full regression or integration testing could cause hidden bugs or errors impacting production users.

Rollback plan if this doesn't work:

1. Revert to previous dbt model version:

Restore the last known good version of the `analytics.fct\_orders` model that uses the join to `customers` (old method for `region\_code`) in the production dbt project.

2. Re-run dbt on production:

Run dbt to recreate the mart using the reverted model, ensuring the pipeline and dependent downstream assets return to a stable state.

3. Communicate rollback:

Inform key stakeholders and downstream consumers about the rollback and temporary reversion to old logic until the schema migration is fully ready.

4. Coordinate schema migration completion:

Work with the data engineering or DBA teams to finalize the column deployment, backfill missing data if needed, and validate before attempting the migration again.

5. Post-rollback monitoring:

Closely monitor production metrics, error logs, and data quality checks to confirm system stability and correctness after rollback.

This rollback approach minimizes production impact by reverting only the problematic model logic, allowing time to address underlying schema readiness without blocking critical data deliveries.

Engineering counterpart
7:12 AM - Responded in 0s

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

Agent response

7:12 AM - Responded in 9s

Certainly. Here's a concrete next steps plan with clear owners and timing:

Immediate Next Steps (Within 2 hours)

1. Validate `region\_code` column in source `orders` table
 - Owner: Data Engineering Lead
 - Action: Run schema inspection and data profiling queries on production `orders` table to confirm `region\_code` exists and is populated.
 - Deliverable: Confirmation report (yes/no + data quality notes)
2. Check and update staging model `stg\_orders`
 - Owner: Data Engineering Team (Modeling Specialist)
 - Action: Review and update `stg\_orders` model to ensure it sources `region\_code` directly from `orders`, aligned with the new schema.
 - Deliverable: Updated `stg\_orders` model code, tested in dev/staging environment.
3. Run full dbt run in dev/staging environment
 - Owner: Data Engineering Team
 - Action: Compile and run the `fct\_orders` model and dependent models locally or in staging with production-like data.
 - Deliverable: Successful run with zero errors, validated outputs.

Follow-up Steps (Within 1 business day)

4. Code review and data validation
 - Owner: Analytics Engineering & QA Team
 - Action: Review updated models, compare key metrics vs. previous runs, especially for `region\_code` correctness.
 - Deliverable: Signed-off data validation report.
5. Stakeholder communication and deployment planning
 - Owner: Data Engineering Manager / Product Owner
 - Action: Inform analytics, BI teams, and business stakeholders about upcoming fix, timeline, and rollback plans.
 - Deliverable: Communication sent, feedback collected.
6. Production deployment with monitoring
 - Owner: Data Engineering Release Engineer
 - Action: Deploy updated models and run full dbt pipeline in production outside business-critical hours if possible. Monitor logs and data quality dashboards.
 - Deliverable: Successful production run or immediate rollback trigger.

Rollback Plan (Ready to execute immediately if needed)

- Owner: Data Engineering Release Engineer
- Action: Revert `fct\_orders` to last stable version using previous join approach, run full dbt production run, notify stakeholders.
- Timing: Within 1 hour of rollback decision.

Summary:

By end of current day, confirm schema readiness, update staging models, test thoroughly, communicate plan, and deploy with monitoring. Have rollback ready to minimize downtime or data issues.

Engineering counterpart**7:12 AM - Responded in 0s**

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

Agent response**7:12 AM - Responded in 3s**

The one thing I'd want a second pair of eyes on is the alignment between the updated `fct\_orders` model SQL and the actual physical schema of the `orders` source and `stg\_orders` staging table, specifically verifying that `region\_code` is correctly present, typed, and accessible where the model expects it.

This includes:

- Reviewing the compiled SQL referencing `O.REGION\_CODE` to ensure aliasing and column names are correct.
- Confirming that the `stg\_orders` model reflects the new schema and exposes `region\_code` properly.
- Validating that no lingering references to the old join logic remain in the codebase that could cause partial or inconsistent failures.

A careful peer review here helps catch subtle mismatches or assumptions that can cause the same compilation error or downstream data correctness issues post-deployment.

Engineering counterpart**7:12 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.