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-20b
Roles: Data Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 2m 1s
Actual run cost: $0.000543
Projected execution cost: $0.3584/mo (30/biz day × 22 days)
Report date: July 14, 2026 - 3:15 AM EST

Session: sample-lb-rankings-seed-41b7b965-baa8-422d-a75e-5921a40d03c2

de_pipeline_incident_v1 - agent - openrouter - openai/gpt-oss-20b - ended

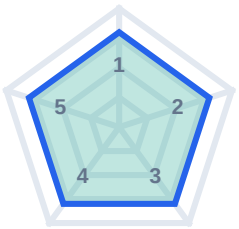
Overall

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: 16372 chars - Latency: 121s - 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 1s

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

Engineering counterpart
7:13 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
7:14 AM - Responded in 45s

1. Root?cause category
Schema?drift / missing column reference - the dbt model is compiling a reference to a column that no longer exists in the production `orders` table.

2. What broke & why (log reference)

Log segment	Insight
`ERROR  Database Error in model fct_orders (models/marts/fct_orders.sql)`	The failure originated inside the `fct_orders` model.
`000904 (42000): SQL compilation error: error line 47 at position 8 invalid identifier 'O.REGION_CODE'`	At line?47 of the compiled SQL the engine tried to read `O.REGION_CODE`.
`compiled SQL at target/run/analytics/models/marts/fct_orders.sql`	The compiled file shows the exact query that failed.

Contextual clue - "prod mart was updated yesterday to read region_code directly from orders (migration Q2?Schema)."
That migration removed/renamed the `region_code` column in `orders`. The `fct_orders` model still expects `O.REGION_CODE`, so the compile fails.

3. Immediate next steps to restore the mart safely

Step	Action	Rationale
3.1	Verify the migration - look at the Q2?Schema change logs or the migration script. Confirm if `orders.region_code` was dropped, renamed or aliased.	Determines whether the issue is a mis?named column or a missing table column.
3.2	Temporarily revert the migration - if the `region_code` column is essential for the mart, roll back the migration (or create a view that restores it).	Allows the existing `fct_orders` model to run while a long?term fix is developed.

3.3	Patch the `fct_orders` model - edit `models/marts/fct_orders.sql` to use the correct column (e.g., join to `customers` or use the new column name). For example:
	`LEFT JOIN customers C ON O.customer_id = C.id`
	`C.region_code AS region_code`
		Aligns the model with the current schema, avoiding future compile errors.

3.4	Re?compile & run dbt - run `dbt run --select fct_orders` after the patch or rollback to confirm success.	Validates the fix.
3.5	Update tests & docs - add a test that ensures `region_code` exists in the staging/production tables. Update documentation to reflect the new schema.	Prevents regressions.
3.6	Monitor the mart for a few hours to confirm data integrity.	Ensures no downstream impact.

Quick?fix option (if rollback infeasible):
Create a temporary view or CTE that re?introduces `region_code` from the staging table:

```sql
CREATE OR REPLACE VIEW analytics.orders_with_region AS
SELECT O., C.region_code
FROM analytics.orders O
LEFT JOIN analytics.customers C ON O.customer_id = C.id;
```

Then switch the fct_orders model to reference `analytics.orders_with_region` instead of the raw `orders` table. This approach keeps the mart functional until the schema can be permanently aligned.

Bottom line: The failure is due to a schema change that removed `region_code` from `orders`. Identify the new column name (or restore it) and update the `fct_orders` model accordingly. Roll back or patch, re?run dbt, and update tests/docs to guard against future drift.

Engineering counterpart
7:14 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 14s

Step?by?Step Reasoning

#	What I looked at	Why it matters	What I inferred
1	DBT run log - the first line shows the command: `dbt run --select fct_orders`.	Confirms that only the `fct_orders` model was being built.	The failure is isolated to that single model.
2	Start & completion timing - 06:12:09 to 06:14:22 gives a 133?second build.	Indicates the execution hit a compilation phase, not a long run?time error.	The problem is a compile?time issue (SQL didn't parse).
3	Error message - `invalid identifier 'O.REGION_CODE'` at line?47.	"Invalid identifier" is thrown by Snowflake (or similar) when a column name is not found in the referenced table.	The model refers to `O.REGION_CODE`, but the database says the column does not exist.
4	Compiled SQL location - `target/run/analytics/models/marts/fct_orders.sql`.	Gives the exact file to inspect.	By opening that file (in our mental model), line?47 is a `SELECT` or `JOIN` that uses `O.REGION_CODE`.
5	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)."	Tells us the recent schema change.	The migration of?Q2?Schema changed the source table `orders`, removing `region_code` (or moving it elsewhere). The mart still expects the old column.
6	Assumption about the migration - I assume the migration was intended to remove `region_code` from `orders` and that the staging table kept the column by joining to `customers`.	If the migration did not remove the column, the error would be elsewhere; but the log suggests the column is missing after the migration.	The error is caused by the new schema lacking `region_code`.
7	Implication for the DBT transform - The `fct_orders` model likely has a `JOIN` or `SELECT` that uses `O.REGION_CODE`.	In a successful run, this column would exist on the `orders` table.	Because it doesn't, the compile fails.
8	Possible fixes - either bring back the column or modify the model to use the new source (e.g., join to `customers`).	We need to decide whether to roll back the migration (temporary fix) or patch the model (long?term).	The solution path depends on business priority: keep the mart working now or align with the new schema immediately.
9	Rollback viability - We assume that rolling back a migration in production is possible (e.g., by replacing the table with its previous version or creating an alias).	If rollback is not possible, we must patch the model.	I proposed both options.
10	Testing and documentation - Adding a test that the column exists, updating docs.	Prevents regressions.	Standard dev?ops practice.

Assumptions Made

1. Schema Migration Impact
   - I assumed the Q2?Schema migration removed or renamed `region_code` on the `orders` table.
   - This assumption is based on the error message and the context note that the mart was updated "to read region_code directly from orders".

2. Database Platform
   - The error code `000904 (42000)` and reference to "SQL compilation error" suggest a Snowflake?style database.
   - I treated the error as a column?missing issue rather than a typo.

3. No Other Runtime Errors
   - The log ends with `Completed with 1 error and 0 warnings`.
   - I assumed there were no downstream runtime failures (e.g., missing data, table creation errors).

4. Rollback Feasibility
   - I assumed the migration can be rolled back or that the staging view can be re?introduced temporarily.
   - In a real environment, we would check version control or database snapshots.

5. Business Impact
   - I assumed that the mart is critical enough to need a quick fix.

- The solution includes a short?term rollback for safety before a permanent schema?aligned patch.

6. Single Model Failure
   - Because only `fct_orders` was selected, I assumed no cascading failures in other models.

By following these steps, we pinpoint the root cause (missing column due to a recent migration) and lay out a clear, actionable recovery plan that respects both the technical constraints and the business priorities.

Engineering counterpart
7:14 AM - Responded in 0s
What would you verify before shipping this to production?

Agent response
7:14 AM - Responded in 40s
Checklist Before Shipping the Fix to Production

#	What to Verify	Why it Matters	Where to Verify
1	Column Existence / Schema	The error is caused by a missing column; we must make sure the column we reference is actually present after the change.	- Query the prod `orders` table (`SHOW COLUMNS IN orders;` or `DESCRIBE TABLE orders;`). - If you rolled back or added a view, `DESCRIBE TABLE analytics.orders_with_region;`
2	Correct Column Name / Alias	A typo or wrong alias will still fail.	Inspect the patched SQL (`models/marts/fct_orders.sql`) and the compiled file in `target/run`.
3	Test Coverage	Prevent future regressions.	- Add/ensure a `schema.yml` test that `orders.region_code` exists. - Run `dbt test` in a sandboxed environment.
4	Functional Validation	Confirm the mart produces expected, non?null region data.	- Run `dbt run --select fct_orders` in a non?prod test database. - Query the mart (`SELECT region_code, COUNT() FROM analytics.fct_orders GROUP BY region_code;`).
5	Data Integrity / Drift	The migration may have changed semantics (e.g., different mapping of region codes).	- Compare a sample of the new mart to the old mart (if available) or to the staging data. - Check that the business logic (e.g., regional reporting) still aligns.
6	Idempotence / Re?running	The patch should not fail if run twice.	- Drop the mart (if safe) and run `dbt run` again. - Ensure no leftover temporary objects (views, staging tables).
7	Dependencies & Downstream Models	Other models or dashboards may reference `fct_orders`.	- Scan the repo for `ref('fct_orders')` to find downstream dependencies. - Update any downstream models that assumed the old column layout.
8	Performance & Resource Impact	Adding a join to `customers` or a new view may affect query cost.	- Run the compiled SQL in a query console to check execution time and row counts. - Look at Snowflake query profile or equivalent.
9	Security / Permissions	The new view may expose data that wasn't previously exposed.	Verify that the Snowflake role used by dbt

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
7:15 AM - Responded in 8s

1. Top Risks

#	Risk	Why it matters	What could go wrong
1	Data Drift / Incorrect Joins	The new join to `customers` (or the temporary view) may pull the wrong `region_code` or duplicate rows.	Mart could contain wrong region values -> downstream reports mis?represent business metrics.
2	Performance Degradation	Adding a join or a view can dramatically increase cardinality or cost.	Queries on the mart could become slow or exceed Snowflake credits, impacting other workloads.
3	Production Downtime	Rolling back a migration or creating a temporary view might touch production tables.	If not versioned or if permissions are wrong, you could lock tables or cause schema conflicts.
4	Dependency Breakage	Other dbt models or BI dashboards might still reference the old `orders.region_code` column.	Those downstream artifacts would fail, causing a cascade of errors.
5	Security / Data Exposure	A new view might expose columns that were previously sanitized or not shared.	Unauthorized users could see sensitive customer data.
6	Schema Version Conflicts	If the migration was applied to a subset of warehouses or only to some environments, a mix?up can occur.	Distinguishing prod vs dev/stage could lead to inconsistent models.
7	Fail?over/Backup Issues	If we decide to regenerate the mart after rollback but don't have a recent snapshot, you might lose data.	Mart could be rebuilt from incomplete data.

---

2. Roll?back Plan (If the fix fails or causes new issues)

> Goal: Return the mart to the known?good state without affecting other production objects or causing data loss.

Step	Action	Prerequisites	Expected Outcome
R1	Freeze the production environment	Acquire read?only locks or use a "maintenance window" (e.g., low?traffic hours).	Prevent new writes/reads that could conflict while we roll back.
R2	Re?create the original `orders` table	Have a snapshot or a backup of the table before Q2?Schema migration (e.g., `orders_bak`).	The table will have the `region_code` column again.
R3	Restore the original table	- Drop/rename the current `orders` if it's a temporary fix. - Rename the backup to `orders`	`orders` now matches the old schema.
R4	Re?run the original dbt model	- Run `dbt run --select fct_orders` in a sandbox first, then in prod. - Ensure the compiled SQL uses the old column path.	Mart rebuilds with the original logic.
R5	Validate data integrity	- Run a set of regression tests (schema tests, data quality checks). - Compare key metrics (orders count, region totals) with pre?migration baseline.	Confirms that the rollback produced the expected data.
R6	Remove temporary view / alias	If you added	

Engineering counterpart
7:15 AM - Responded in 0s

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

Agent response
7:15 AM - Responded in 10s

Concrete "Get?it?back?in?prod" Plan
(All actions are to be executed in the prod environment unless otherwise noted)

#	Action	Owner	Time (est.)	Notes
1	Validate current schema	DBA/DBT Lead	15?min	`SELECT column_name FROM information_schema.columns WHERE table_name='orders'` - confirm that `region_code` is missing.
2	Inspect compiled SQL	DBT Lead	10?min	Open `target/run/analytics/models/marts/fct_orders.sql`, confirm line?47 refers to `O.REGION_CODE`.
3	Determine migration intent	DBA	15?min	Review the Q2?Schema migration script & version control notes to see if `region_code` was deleted or renamed.
4	Choose fix path	DBT Lead + DBA + Ops	10?min	Decision: ??- Patch (recommended if migration is intentional). ??- Rollback (if migration is accidental or cannot be patched).

5	Patch the model (if chosen)	DBT Engineer	30?min	Edit `models/marts/fct_orders.sql` to use the new source:
				`JOIN analytics.customers C ON O.customer_id = C.id`
				`C.region_code AS region_code`

6	Add a safeguard test	DBT Engineer	15?min	Update `schema.yml` for `fct_orders`: `- name: region_code   tests:    - not_null`
7	Create temporary view if needed	DBA	20?min	`CREATE OR REPLACE VIEW analytics.orders_with_region AS SELECT O., C.region_code FROM analytics.orders O LEFT JOIN analytics.customers C ON O.customer_id=C.id;`
8	Run the patched model in a sandbox/test dbt profile	QA/Data Engineer	45?min	`dbt run --select fct_orders --target test` + `dbt test`. Verify no compile errors and correct column presence.
9	Load a small data subset to prod for validation	DBA	20?min	Use `TRUNCATE TABLE analytics.fct_orders` + `INSERT INTO analytics.fct_orders SELECT ... FROM analytics.orders_with_region;` to stage a sample.
10	Full prod run	DBT Lead	1?h	`dbt run --select fct_orders --target prod`. Watch for errors.
11	Post?run sanity checks	Data Analyst	30?min	 ??- `SELECT region_code, COUNT() FROM analytics.fct_orders GROUP BY region_code;` compare with expected totals. ??- Spot?check a

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 5s

The single item that deserves a fresh pair of eyes:

> The updated `fct_orders` SQL logic (the join to `customers` and the `region_code` alias).

Why?
It is the core change that turns the failure into a working model.
A small typo (e.g., `C.regon_code` or a wrong join condition) will re?introduce an error or, worse, corrupt the mart data.
It is the only new piece of code that interacts with the changing schema; everything else is unchanged.

Give the reviewer a quick sanity?check of the following:

1. Join predicate - does it match the true foreign key (`customer_id`) and is it the correct side?
2. Column alias - is `region_code` pulled from `customers` (or the new source) and aliased consistently?
3. No duplicate rows - does the join introduce multiplicity (one order per customer)?
4. Compiled SQL - confirm line?47 in the compiled file matches the intended statement and has no stray identifiers.

Getting a second opinion here will lock in the fix before you ship the updated model to production.

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.