



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: qwen/qwen3-32b

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

Duration: 1m 27s

Actual run cost: \$0.001308

Projected execution cost: \$0.8633/mo (30/biz day × 22 days)

Report date: July 14, 2026 - 3:16 AM EST

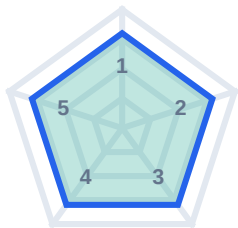
Overall

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: 8667 chars - Latency: 87s - 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: 1m 27s

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

Engineering counterpart

7:15 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

7:15 AM - Responded in 17s

(1) Root Cause Category: Schema mismatch (data model drift from source system schema).

(2) What Broke:

- The error `invalid identifier 'O.REGION\_CODE'` indicates the `fct\_orders` model references a `region\_code` column in the `orders` source table that does not exist in the source schema.
- The log context confirms that the source system still exposes `region` via a

Engineering counterpart

7:15 AM - Responded in 0s

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

Agent response

7:15 AM - Responded in 17s

(1) Step-by-Step Reasoning

- Error Line Analysis:

The log explicitly states 'invalid identifier 'O.REGION_CODE'' (line 47). This means the model 'fct_orders' is trying to reference a column 'region_code' in an 'orders' source table, but the database does not recognize this column.

- Assumption: If the database throws a compilation error about an invalid identifier, the column likely doesn't exist in the source schema (e.g., raw table or a staging table used in 'fct_orders').

- Context Validation:

The note clarifies:

- The staging table 'stg_orders' still exposes 'region' via a join to 'customers' (implying 'region_code' is derived from another table).

- The prod mart was updated yesterday to read 'region_code' directly from 'orders' as part of a schema migration.

- Assumption: The 'orders' table in the source system (likely a raw table) does not have a 'region_code' column, but the migration in the data model assumed it existed. This creates a schema drift between the model and the source.

- Root Cause Deduction:

- The 'fct_orders' model now references 'O.REGION_CODE' (presumably as a field in the raw 'orders' table).

- However, the raw source (e.g., 'raw_orders' or the source of 'stg_orders') still requires joining with 'customers' to get region info.

- The schema migration in the model ('migration Q2-Schema') incorrectly assumed 'region_code' is directly available in 'orders' when it is not.

(2) Assumptions Made

- Source Schema: The raw 'orders' table does not contain 'region_code'. Instead, region data is derived via a join with 'customers' in the staging layer.

- Model Dependency: The 'fct_orders' model

Engineering counterpart

7:15 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:15 AM - Responded in 16s

Pre-Production Verification Steps

Before shipping a fix to production, I'd verify the following to ensure the root cause is fully resolved and no new issues are introduced:

1. Confirm Source Schema Drift

- What to check:

- Does the raw/staging 'orders' table actually lack a 'region_code' column?

- Is the 'region' field derived via a join to 'customers' (as per the staging logic)?

- How:

- Run a schema validation query (e.g., 'DESCRIBE TABLE raw_orders' or 'INFORMATION_SCHEMA.COLUMNS') to confirm the absence of 'region_code'.

- Validate the 'stg_orders' model's SQL to ensure it uses a join with 'customers'

Engineering counterpart

7:15 AM - Responded in 0s

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

Agent response

7:16 AM - Responded in 10s

Top Risks

1. Data Inaccuracy Due to Schema Drift:

- Even after fixing the `region\_code` reference, the join logic used to derive it from `orders` and `customers` might not match the schema of the raw source tables (e.g., mismatched keys like `customer\_id` or missing relationships).
- Risk: Incorrect joins could lead to missing or invalid region codes in the `fct\_orders` model, breaking downstream dashboards/marts.

2. Performance Degradation:

- Adding a join to `customers` in `fct\_orders` could increase query complexity, especially if the `orders` or `customers` tables are large.
- Risk: Slower query performance might delay downstream workflows or fail under load.

3. Unintended Side Effects from Schema Changes:

- If

Engineering counterpart

7:16 AM - Responded in 0s

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

Agent response

7:16 AM - Responded in 14s

Concrete Next Steps (with Owners & Timing)

1. Verify Root Cause (Owner: Data Engineer)

- Action:
 - Check the source `orders` table schema (e.g., `DESCRIBE RAW\_ORDERS` or `SHOW COLUMNS FROM ORDERS`) to confirm `region\_code` does not exist.
 - Validate the `stg\_orders` model to ensure it correctly joins `orders` and `customers` to derive `region\_code`.
- Timing: 15 minutes.

2. Schema Validation (Owner: Data Model Team)

- Action: Confirm whether the Q2-Schema migration was applied to the source system. If not, the source schema drift explains the error.
- Timing: 30 minutes.

3. Fix Dependency in `fct\_orders`

- Action:
 - Revert `fct\_orders` to join with `stg\_orders` (not raw `orders`) to source `region\_code` as before.
 - Remove any hardcoded logic assuming `region\_code` exists in raw `orders`.
- Timing: 1 hour (including testing in dev/staging).

4. Data Validation (Owner: Data Quality Team)

- Action:
 - Run post-fix validation tests (e.g., compare `fct\_orders.region\_code` output to historical data or upstream dashboards).
 - Ensure the join logic produces consistent results (no nulls/changes in `region\_code` distribution).
- Timing: 1 hour.

5. Performance Testing (Owner: Data Operations Team)

- Action:
 - Benchmark the updated `fct\_orders` query against the previous version (pre-Q2-Schema migration) to ensure no significant performance degradation.
 - Optimize joins or add indexes if needed.
- Timing: 2 hours.

6. Production Rollout (Owner: Data Engineer)

- Action:
 - Schedule a staged deployment (e.g., canary run) to ensure stability.
 - Monitor the next dbt run for success (target=prod) and downstream consumers.
- Timing: 1 hour.

Rollback Plan

If the fix fails or introduces errors in production:

1. Immediate Reversion (15 mins):

- Revert the `fct\_orders` model to its previous valid state (pre-Q2-Schema migration) using version control (e.g., Git checkout).
- Retry the dbt run with the reverted model to restore the mart.

2. Temporary Workaround (30 mins):

- If the schema migration is incomplete, temporarily patch the source (e.g., backfill `region\_code` in `orders` via a one-time view or CTE) until the migration is finalized.

3. Post-Rollback Root Cause Investigation:

- Collaborate with the Data Model Team to resolve schema drift (e.g., confirm if `region\_code` should exist in `orders` or fix the migration).
- Update `fct\_orders` only after the source schema is consistent.

Key Assumptions

- The `orders` table in the source system (raw data) never had a `region\_code` field.
- The `stg\_orders` model correctly exposes `region\_code` via its existing join logic.
- The schema migration (Q2-Schema) was not fully applied to the source system, causing the discrepancy.

Engineering counterpart

7:16 AM - Responded in 0s

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

Agent response

7:16 AM - Responded in 13s

Critical Step for Peer Review

The join logic in `stg\_orders` to derive `region\_code` from the `customers` table (specifically the join condition and key alignment between `orders` and `customers`).

Why This Needs a Second Pair of Eyes

- Risk: The `fct\_orders` model now depends on `stg\_orders` to provide accurate `region\_code` values. If the `stg\_orders` join logic (e.g., join on `customer\_id`, handling of nulls, or data mapping) is flawed, the fix to revert `fct\_orders` could propagate incorrect or incomplete region data to downstream consumers.
- Example: If `stg\_orders` joins `orders.customer\_id` to `customers.id` but the source systems use different naming (e.g., `customer\_key` vs. `id`), the `region\_code` will be invalid.
- Impact: This could silently reintroduce the same data quality issues the schema migration was intended to resolve.

What to Validate

- Join Keys: Confirm that `stg\_orders` correctly maps `orders.customer\_id` to `customers.customer\_id` (or the correct identifier).
- Data Coverage: Ensure the join does not create unintended nulls in `region\_code` (e.g., due to missing `customer\_id` in `orders`).
- Schema Alignment: Verify that the `customers` table used in `stg\_orders` has the `region\_code` column as expected post-migration.

Owner & Timing

- Owner: Data Model Team (specifically the person who owns the `stg\_orders` model).
- Timing: 30 minutes before merging the fix to production, to avoid compounding errors.

Rollback Plan Integration

If the peer review uncovers issues in `stg\_orders`:

1. Pause the fix to `fct\_orders`.
2. Correct the staging layer first (e.g., fix join keys or ensure `customers` has valid `region\_code`).
3. Revalidate the full pipeline (`stg\_orders` -> `fct\_orders` -> downstream dashboards) before proceeding.

This ensures the fix in `fct\_orders` is not masking a deeper issue in the staging layer.

Engineering counterpart

7:16 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.