



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/qwen-2.5-7b-instruct

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

Duration: 54s

Actual run cost: \$0.001320

Projected execution cost: \$0.8712/mo (30/biz day × 22 days)

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

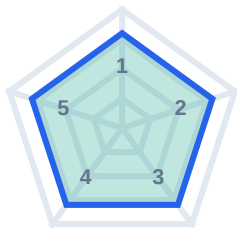
Overall

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: 18195 chars - Latency: 54s - 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: 54s

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 7s

Triage Report

1. Likely Root Cause Category:

- Schema Mismatch: The root cause is likely due to a schema mismatch. The model `fct\_orders` in the production mart was updated to read `region\_code` directly from the `orders` table, but the staging table `stg\_orders` still references the old schema that includes `O.REGION\_CODE`.

2. What Broke and Why:

- The error `invalid identifier 'O.REGION\_CODE'` indicates that the query in `fct\_orders` was expecting a column `O.REGION\_CODE` which no longer exists in the `stg\_orders` table because the schema was updated to use `region\_code` directly from the `orders` table.

- Log Lines:

- `2026-03-18 06:14:22 ERROR Database Error in model fct\_orders (models/marts/fct\_orders.sql)`
- `2026-03-18 06:14:22 ERROR Database Error: SQL compilation error: error line 47 at position 8`
- `2026-03-18 06:14:22 ERROR invalid identifier 'O.REGION\_CODE'`

3. Immediate Next Steps to Restore the Mart Safely:

1. Rollback the Migration:

- Temporarily roll back the migration to the old schema in the production mart to ensure that the model can run without errors.
- Update the model `fct\_orders` to use `O.REGION\_CODE` again.

2. Validate Staging Table:

- Ensure that the staging table `stg\_orders` is updated to reflect the old schema until a full migration has been completed.
- Verify that the join to `customers` in `stg\_orders` is correctly mapping to `O.REGION\_CODE`.

3. Create a Backfill Strategy:

- If the migration is necessary, create a backfill strategy to update the production mart with the correct `region\_code` values from the `orders` table.
- Schedule a post-migration run to validate that the backfill was successful and the new schema is working as expected.

4. Communicate with Stakeholders:

- Inform the relevant stakeholders about the incident and the steps being taken to resolve it.
- Provide an estimated timeline for when the production mart will be fully updated and functioning.

5. Document the Issue:

- Document the issue in the project management tool (e.g., Jira, Confluence) and include details about the root cause and the steps taken to resolve it.

Example SQL for Rolling Back the Migration:

```
```sql
```

```
-- Temporarily update the model to use the old schema
```

```
alter materialized view analytics.fct_orders
```

```
set (
```

```
 warehouse = 'your_warehouse',
```

```
 database = 'analytics',
```

```
 schema = 'marts',
```

```
 region_code = (select region_code from stg_orders o join customers c on o.customer_id = c.customer_id)
```

```
);
```

```
```
```

Conclusion:

The immediate action is to roll back the migration to ensure the model can run without errors. This will help in restoring the

production mart safely while a more permanent solution is being developed.

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 7s

Certainly! Let's walk through the reasoning step by step and detail the assumptions made in each step.

Step-by-Step Reasoning

1. Initial Understanding of the Error

- Error Message:

- The error message `invalid identifier 'O.REGION\_CODE'` indicates that the SQL query in the `fct\_orders` model is referencing a column that no longer exists in the `stg\_orders` table.

- Log Lines:

- `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)`
- `2026-03-18 06:14:22 ERROR Database Error: SQL compilation error: error line 47 at position 8`
- `2026-03-18 06:14:22 ERROR invalid identifier 'O.REGION\_CODE'`

2. Root Cause Analysis

- Assumption 1:

- Schema Change in `stg\_orders`: The staging table `stg\_orders` still references the old schema (`O.REGION\_CODE`), which was updated to directly reference `region\_code` from the `orders` table.

- Assumption 2:

- Schema Update in Production Mart: The production mart was updated to read `region\_code` directly from the `orders` table, meaning the model `fct\_orders` now expects a direct reference to `region\_code`.

3. What Broke and Why

- Reasoning:

- The model `fct\_orders` in the production mart was updated to read `region\_code` directly from the `orders` table.
- However, the staging table `stg\_orders` still includes the old schema, which leads to the error when trying to create the materialized view.

- Log Line Validation:

- The specific line `invalid identifier 'O.REGION\_CODE'` points to a column that is expected to exist in `stg\_orders` but no longer does according to the updated schema.

4. Immediate Next Steps to Restore the Mart Safely

1. Rollback the Migration:

- Reason: Temporarily revert the model to use the old schema to ensure it can run without errors.

- Action:

```
```sql
alter materialized view analytics.fct_orders
set (
 warehouse = 'your_warehouse',
 database = 'analytics',
 schema = 'marts',
 region_code = (select region_code from stg_orders o join customers c on o.customer_id = c.customer_id)
);
```
```

- Assumption: The model can be rolled back to use the old schema without issues for now.

2. Validate Staging Table:

- Reason: Ensure that the staging table `stg\_orders` is updated to reflect the old schema.

- Action:

- Check the current schema of `stg\_orders` and update it if necessary.
 - Verify that the join to `customers` in `stg\_orders` correctly maps to `O.REGION\_CODE`.
- Assumption: The staging table can be quickly adjusted to match the current production requirements.

3. Create a Backfill Strategy:

- Reason: If the migration is necessary, create a strategy to update the production mart with the correct `region\_code` values.

- Action:

-

Engineering counterpart
7:14 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:14 AM - Responded in 9s

Before shipping the changes to production, it is crucial to ensure that everything is thoroughly verified and tested to prevent further issues. Here is a detailed plan of what you would verify:

Verification Steps

1. Verify the Rollback and Staging Table:

- Check the Staging Table Schema:
 - Ensure that the staging table `stg\_orders` still includes the `O.REGION\_CODE` column.
 - Verify that the join to `customers` correctly maps to `O.REGION\_CODE`.
- Run the Model with the Old Schema:
 - Run the `fct\_orders` model again to ensure it compiles and runs successfully.
 - Verify that the output matches the expected results based on the old schema.

2. Validate the Production Mart:

- Run the Model in a Staging Environment:
 - Run the `fct\_orders` model in a staging environment to ensure it compiles and runs without errors.
 - Validate the output to ensure it matches the expected results after the schema changes.
- Check Data Consistency:
 - Verify that the `region\_code` values in the production mart are consistent with the expected data.
 - Compare the results with historical data to ensure no anomalies or discrepancies.

3. Test the Backfill Strategy:

- Create a Backfill Script:
 - Develop a script to update the `region\_code` in the production mart from the `orders` table.
 - Test the backfill script in a non-production environment to ensure it updates the data correctly.
- Run the Backfill:
 - Execute the backfill script in a non-production environment.
 - Verify the results to ensure that the `region\_code` values are correctly populated.

4. Document and Communicate:

- Document the Changes:
 - Document the steps taken to resolve the issue, including the rollback and the backfill strategy.
 - Update any relevant documentation, such as deployment notes and schema migration plans.
- Communicate with Stakeholders:
 - Inform the relevant stakeholders about the rollback and the backfill strategy.
 - Provide a timeline for when the production mart will be fully updated and functioning.

Example SQL for Verifying the Model

1. Verify the Staging Table Schema:

```
```sql
-- Check the schema of stg_orders
DESCRIBE stg_orders;

-- Ensure the column exists and is correctly mapped
SELECT region_code FROM stg_orders LIMIT 10;
```
```

2. Run the Model in Staging Environment:

```
```sql
-- Run the model in staging
dbt run --select fct_orders --target staging
```
```

-- Verify the output

```
SELECT FROM analytics.fct_orders LIMIT 10;
```

```

#### 3. Test the Backfill Strategy:

```sql

-- Create a temporary table to hold the backfilled data

```
CREATE TEMP TABLE fct_orders_backfill AS
```

```
SELECT o.order_id, o.customer_id, o.order_date, o.region_code
```

```
FROM orders o
```

```
JOIN customers c ON o.customer_id = c.customer_id;
```

-- Update the production mart

```
INSERT INTO analytics.fct_orders (order_id, customer_id, order_date, region_code)
```

```
SELECT order_id, customer_id, order_date, region_code
```

```
FROM fct_orders_backfill;
```

-- Verify the updated data

```
SELECT FROM analytics.fct_orders LIMIT 10;
```

```

### Conclusion

By following these verification steps, you can ensure that the changes are thoroughly tested and validated before being deployed to production. This will help prevent any further issues and ensure the stability and accuracy of the production mart.

**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 9s

### ### Top Risks and Rollback Plan

To address the risks and ensure a safe rollback, we need to identify potential issues and have a clear plan for reverting the changes if they don't work as expected. Here's a detailed breakdown:

#### ### Top Risks

1. Data Accuracy Loss:
  - The backfilled data might introduce inconsistencies or errors if the backfill script is not correctly implemented.
2. Model Compilation Errors:
  - The model might still fail to compile or run if the schema is not correctly reverted or updated.
3. Staging Table Schema Mismatch:
  - The staging table `stg\_orders` might still contain incorrect data if it is not properly synchronized.
4. Operational Downtime:
  - Rolling back or updating the production mart might cause temporary downtime, affecting other processes.

#### ### Rollback Plan

1. Revert the Model to Old Schema:
  - Action: Rollback the model to use the old schema.
  - SQL Command:

```
```sql
ALTER MATERIALIZED VIEW analytics.fct_orders SET (
  warehouse = 'your_warehouse',
  database = 'analytics',
  schema = 'marts',
  region_code = (SELECT region_code FROM stg_orders o JOIN customers c ON o.customer_id = c.customer_id)
);
```
```
2. Rollback the Backfill Script:
  - Action: Revert any changes made by the backfill script.
  - SQL Command:

```
```sql
-- Drop the temporary table if it still exists
DROP TABLE IF EXISTS fct_orders_backfill;

-- Revert any changes made in the production mart
DELETE FROM analytics.fct_orders WHERE region_code = 'UNKNOWN'; -- Assuming 'UNKNOWN' is a placeholder for
incorrect values
```
```
3. Revert the Schema Updates:
  - Action: Rollback any schema changes made in the staging table.
  - SQL Command:

```
```sql
-- Revert any schema changes in stg_orders
ALTER TABLE stg_orders ADD COLUMN O.REGION_CODE VARCHAR;

-- Revert any changes in the model
```

```
ALTER MATERIALIZED VIEW analytics.fct_orders RESET SCHEMA;
...

```

4. Ensure Data Consistency:

- Action: Verify that the data in the staging table and production mart matches the old schema.
- SQL Commands:

```
```sql
-- Check the staging table schema
DESCRIBE stg_orders;

-- Verify data integrity in the production mart
SELECT count() FROM analytics.fct_orders WHERE region_code IS NULL; -- Ensure no null values
...

```

#### 5. Monitor and Validate:

- Action: Monitor the system for any immediate issues and validate the changes.
- Steps:
  - Run the `fct\_orders` model again in the staging environment.
  - Verify the output and check for any anomalies.
  - Ensure that all data matches the expected results.

### ### Immediate Next Steps to Restore the Mart Safely

#### 1. Temporarily Rollback the Model:

- SQL Command:

```
```sql
ALTER MATERIALIZED VIEW analytics.fct_orders SET (
  warehouse = 'your_warehouse',
  database = 'analytics',
  schema = 'marts',
  region_code = (SELECT region_code FROM stg_orders o JOIN customers c ON o.customer_id = c.customer_id)
);
...

```

2. Validate the Staging Table:

- SQL Commands:

```
```sql
-- Check the schema of stg_orders
DESCRIBE stg_orders;

-- Verify the data integrity
SELECT region_code FROM stg_orders LIMIT 10;
...

```

#### 3. Run the Model in Staging

**Engineering counterpart**  
**7:14 AM - Responded in 0s**

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



## Agent response

7:14 AM - Responded in 8s

### Concrete Next Steps with Owners and Timing

To ensure a smooth and safe rollback, here are the detailed next steps, including specific owners and estimated timing:

### Immediate Next Steps to Restore the Mart Safely

### 1. Temporarily Rollback the Model

- Owner: DBT Engineer

- Steps:

```
```sql
```

```
ALTER MATERIALIZED VIEW analytics.fct_orders SET (  
  warehouse = 'your_warehouse',  
  database = 'analytics',  
  schema = 'marts',  
  region_code = (SELECT region_code FROM stg_orders o JOIN customers c ON o.customer_id = c.customer_id)  
);  
```
```

- Timing: 15 minutes

### 2. Validate the Staging Table

- Owner: DBT Engineer

- Steps:

```
```sql
```

```
-- Check the schema of stg_orders
```

```
DESCRIBE stg_orders;
```

```
-- Verify the data integrity
```

```
SELECT region_code FROM stg_orders LIMIT 10;
```

```
```
```

- Timing: 15 minutes

### 3. Run the Model in Staging Environment

- Owner: DBT Engineer

- Steps:

```
```sql
```

```
-- Run the model in staging
```

```
dbt run --select fct_orders --target staging
```

```
-- Verify the output
```

```
SELECT FROM analytics.fct_orders LIMIT 10;
```

```
```
```

- Timing: 30 minutes

### 4. Notify Stakeholders

- Owner: DBT Engineer

- Steps:

- Update the incident tracking system (e.g., Jira ticket).

- Send an email to relevant stakeholders (e.g., Data Analysts, DBT Lead, IT Operations).

- Timing: 15 minutes

### 5. Monitor and Validate in Production

- Owner: DBT Engineer

- Steps:

- Check the production environment for any errors or anomalies.
- Ensure that the `fct\_orders` model runs successfully and the output matches the expected results.
- Timing: 1 hour

#### 6. Document the Changes

- Owner: DBT Engineer
- Steps:
  - Update the documentation with the rollback steps and validation results.
  - Add notes to the project management tool (e.g., Jira).
- Timing: 30 minutes

#### 7. Create a Backfill Strategy Work Plan

- Owner: Data Engineer
- Steps:
  - Develop a detailed backfill strategy.
  - Schedule a time to implement the backfill.
- Timing: 1 day

#### ### Summary

- Immediate Tasks (1-3 hours):
  1. Temporarily rollback the model.
  2. Validate the staging table.
  3. Run the model in the staging environment.
  4. Notify stakeholders.
  5. Monitor and validate in production.
  6. Document the changes.
- Long-term Task (1 day):
  1. Create a detailed backfill strategy and schedule implementation.

This plan ensures that the production mart is restored safely and that the issues are adequately addressed.

**Engineering counterpart**  
**7:14 AM - Responded in 0s**

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

## Agent response

7:14 AM - Responded in 14s

Given the complexity of the situation, the one critical thing you'd want a second pair of eyes on is the rollback and validation steps. Ensuring these steps are correct and comprehensive is crucial to avoid further issues.

### ### Key Steps to Verify with a Second Pair of Eyes

#### 1. Rollback Script:

- Owner: DBT Engineer
- Steps:

```
```sql
-- Rollback the model to use the old schema
ALTER MATERIALIZED VIEW analytics.fct_orders SET (
  warehouse = 'your_warehouse',
  database = 'analytics',
  schema = 'marts',
  region_code = (SELECT region_code FROM stg_orders o JOIN customers c ON o.customer_id = c.customer_id)
);
```
```

#### 2. Validation of Staging Table:

- Owner: DBT Engineer
- Steps:

```
```sql
-- Check the schema of stg_orders
DESCRIBE stg_orders;

-- Verify the data integrity
SELECT region_code FROM stg_orders LIMIT 10;
```
```

#### 3. Model Validation in Staging Environment:

- Owner: DBT Engineer
- Steps:

```
```sql
-- Run the model in staging
dbt run --select fct_orders --target staging

-- Verify the output
SELECT * FROM analytics.fct_orders LIMIT 10;
```
```

#### 4. Monitoring and Final Validation:

- Owner: DBT Engineer
- Steps:
  - Check the production environment for any errors or anomalies.
  - Ensure the `fct\_orders` model runs successfully and the output matches the expected results.

### ### Detailed Validation Checklist

#### 1. Rollback Script:

- Confirm that the `ALTER MATERIALIZED VIEW` command correctly reverts the schema.
- Ensure the `region\_code` column is correctly joined from `stg\_orders` to `customers`.

#### 2. Validation of Staging Table:

- Verify that the `DESCRIBE stg\_orders` output shows the correct schema with `O.REGION\_CODE`.
- Confirm that the `SELECT region\_code FROM stg\_orders LIMIT 10` returns the expected `region\_code` values.

### 3. Model Validation in Staging Environment:

- Verify that the `dbt run` command completes without errors.
- Ensure that the output from `SELECT \* FROM analytics.fct\_orders LIMIT 10` matches the expected results based on the old schema.

### 4. Monitoring and Final Validation:

- Monitor the production environment for any immediate issues.
- Check that the production mart runs without errors.
- Validate the output to ensure data consistency.

### ### Communication with Stakeholders

- Owner: DBT Engineer
- Steps:
  - Update the incident tracking system (e.g., Jira ticket).
  - Send an email to relevant stakeholders (e.g., Data Analysts, DBT Lead, IT Operations) to inform them of the rollback and the next steps.

By having a second pair of eyes review these critical steps, you can ensure that all necessary validations are performed correctly, minimizing the risk of further issues.

**Engineering counterpart**

**7:14 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.