



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: meta-llama/llama-3.3-70b-instruct

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

Duration: 3m 29s

Actual run cost: \$0.002010

Projected execution cost: \$1.33/mo (30/biz day × 22 days)

Report date: July 14, 2026 - 3:12 AM EST

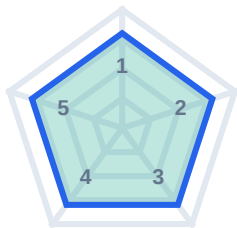
Overall

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: 13870 chars - Latency: 209s - 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: 3m 29s

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

Engineering counterpart

7:09 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

7:09 AM - Responded in 22s

Likely Root Cause Category

The likely root cause category of this failure is a schema mismatch between the production database and the dbt model, specifically related to a recent migration (Q2-Schema) that updated the production mart to read `region\_code` directly from `orders`, but the dbt model or its dependencies still reference the old schema.

What Broke and Why

The error log indicates that the dbt run failed while creating the SQL table model `analytics.fct\_orders`. The specific error message points to a SQL compilation error on line 47 at position 8 of the `fct\_orders.sql` model, complaining about an `invalid identifier 'O.REGION\_CODE'`. This suggests that the model is trying to reference a column `REGION\_CODE` from a table alias `O` (presumably for `orders`), which does not exist or is not accessible as expected.

Given the context, the issue likely stems from the recent migration (Q2-Schema) that changed how `region\_code` is accessed in the production mart. The staging table `stg\_orders` still uses a join to `customers` to expose the region, but the updated mart in production reads `region\_code` directly from `orders`. The dbt model `fct\_orders` seems to be referencing the old schema, leading to the compilation error.

Immediate Next Steps to Restore the Mart Safely

1. Review and Update the dbt Model: Immediately review the `fct\_orders.sql` model and update it to reflect the new schema changes made by the Q2-Schema migration. This involves changing the reference to `O.REGION\_CODE` to correctly access the `region\_code` from the `orders` table according to the updated production schema.
2. Test the Updated Model: Before re-running the dbt production job, test the updated `fct\_orders` model in a development or staging environment to ensure it compiles and runs successfully without errors.
3. Re-run the dbt Job: Once the model has been updated and tested, re-run the failed dbt job to recreate the `analytics.fct\_orders` table with the correct schema.
4. Monitor and Validate: After the job completes, monitor the logs for any further errors and validate that the `fct\_orders` table in the production mart has been correctly updated and is accessible without issues.
5. Documentation Update: Update relevant documentation to reflect the schema changes and the updates made to the dbt models, ensuring that future maintenance and updates can be performed with awareness of the current schema and model dependencies.

Engineering counterpart

7:09 AM - Responded in 0s

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

Agent response

7:10 AM - Responded in 39s

Let's break down the reasoning step by step:

Step 1: Identify the Error Message

The error message is:

```

000904 (42000): SQL compilation error: error line 47 at position 8

invalid identifier 'O.REGION\_CODE'

```

This indicates a SQL compilation error, specifically that the identifier `O.REGION\_CODE` is invalid.

Step 2: Understand the Context

The context provided mentions that:

- The production mart was updated yesterday to read `region\_code` directly from `orders` (as part of the Q2-Schema migration).
- The staging table `stg\_orders` still exposes region via a join to `customers`.

Step 3: Analyze the Error in Context

Given the error message and the context, I inferred that the dbt model `fct\_orders` is attempting to access `REGION\_CODE` from a table alias `O`, which is presumably meant to refer to the `orders` table. However, the error suggests that this reference is invalid.

Step 4: Consider Possible Causes

Based on the information:

- Schema Change: The recent migration changed how `region\_code` is accessed. Before, it might have been accessed through a join with `customers`, but now it's directly from `orders`.
- Model Out of Date: If the dbt model `fct\_orders` has not been updated to reflect this schema change, it would still be trying to access `region\_code` in the old way, leading to the error.

Step 5: Make Assumptions and Draw Conclusions

Assumptions made:

- The `O` alias in the error message refers to the `orders` table.
- The dbt model `fct\_orders` has not been updated to reflect the changes made by the Q2-Schema migration.
- The staging table `stg\_orders` and its method of exposing the region are not directly relevant to the error but indicate how the schema and data flow have been structured before the migration.

Conclusions:

- The likely root cause is a schema mismatch between the updated production database and the dbt model.
- The model needs to be updated to correctly access `region\_code` according to the new schema.

Next Steps

Given these conclusions, the logical next steps involve updating the dbt model to align with the new schema, testing the update, and then re-running the dbt job to apply the changes to the production mart.

Note: The assumptions are based on the provided context and error message. In a real-world scenario, additional information or direct access to the dbt model and database schema might be necessary to confirm these assumptions and ensure the correctness of the conclusions drawn.

Engineering counterpart

7:10 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:10 AM - Responded in 22s

Before shipping the updated dbt model to production, I would verify the following:

1. Correctness of the Update

- Review the changes made to the `fct\_orders` model to ensure that the reference to `REGION\_CODE` is correctly updated to match the new schema.
- Verify that the update does not introduce any new syntax errors or inconsistencies.

2. Compatibility with Dependencies

- Check if any other dbt models or dependencies rely on the old way of accessing `REGION\_CODE` and update them accordingly to avoid cascading errors.
- Ensure that all dependent models are compatible with the changes made to `fct\_orders`.

3. Data Integrity

- Run the updated model against a development or staging environment that mirrors the production environment as closely as possible.
- Validate that the data in the `fct\_orders` table is correctly populated and matches expectations after applying the update.

4. Performance Impact

- Monitor the performance of the dbt job when running the updated model to ensure it does not introduce significant performance degradation.
- Check for any potential bottlenecks or optimization opportunities.

5. Regression Testing

- Perform regression testing to ensure that the update does not break any existing functionality or introduce new errors in other parts of the data pipeline.
- Run a suite of tests that cover various scenarios and edge cases to validate the integrity of the data pipeline.

6. Documentation and Change Management

- Update relevant documentation to reflect the changes made to the dbt model and the new schema.
- Ensure that all stakeholders are informed about the changes, including any potential impact on downstream processes or dependent systems.

7. Rollback Plan

- Have a rollback plan in place in case the update causes unforeseen issues in production.
- Ensure that it's possible to quickly revert to the previous version of the model if necessary.

8. Automated Tests and Checks

- If available, run automated tests and checks (e.g., data validation tests, schema checks) to verify the correctness and integrity of the data after applying the update.

By verifying these aspects, you can ensure a smoother transition to production and minimize the risk of introducing errors or downtime.

Engineering counterpart

7:10 AM - Responded in 0s

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

Agent response

7:11 AM - Responded in 34s

Top Risks

1. Data Corruption: The updated model might incorrectly transform or load data into the `fct\_orders` table, leading to data corruption or inconsistencies.
2. Dependency Issues: Changes to the `fct\_orders` model could break dependencies with other models or downstream processes, causing a ripple effect of errors.
3. Performance Degradation: The updated model might introduce performance issues, such as slower load times or increased resource utilization, affecting the overall data pipeline.
4. Incorrect Results: The changes might produce incorrect results or affect the accuracy of reports, analytics, or other processes relying on the `fct\_orders` table.
5. Downtime: If the update fails or causes significant issues, it might lead to downtime or delays in the data pipeline, affecting business operations or decision-making.

Rollback Plan

In case the update does not work as expected, the rollback plan should include the following steps:

1. Immediate Reversion: Revert the `fct\_orders` model to its previous version, ensuring that any changes made are reversed.
2. Restore Previous State: Restore the `fct\_orders` table to its previous state, using backups or version control, to ensure data integrity and consistency.
3. Re-run Previous Job: Re-run the previous dbt job that successfully loaded data into the `fct\_orders` table, ensuring that the data pipeline is restored to its previous state.
4. Validate Data Integrity: Validate the data integrity of the `fct\_orders` table and any dependent tables or processes, ensuring that the rollback did not introduce any new issues.
5. Notify Stakeholders: Notify all relevant stakeholders, including data consumers, analysts, and business users, about the rollback and any potential impact on their work or decision-making.
6. Post-Rollback Review: Conduct a post-rollback review to identify the root cause of the issue, update the model and testing procedures accordingly, and implement measures to prevent similar issues in the future.

Rollback Strategies

To minimize downtime and data loss, consider the following rollback strategies:

1. Version Control: Use version control systems, such as Git, to track changes to the dbt model and easily revert to previous versions.
2. Backups: Regularly back up the `fct\_orders` table and other critical data assets to ensure quick restoration in case of issues.
3. Staging Environment: Use a staging environment to test and validate changes before deploying them to production, reducing the risk of errors and downtime.
4. Automated Testing: Implement automated testing and validation procedures to detect issues early and prevent them from reaching production.

By having a well-planned rollback strategy in place, you can quickly respond to issues, minimize downtime, and ensure the integrity of your data pipeline.

Engineering counterpart

7:11 AM - Responded in 0s

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

Agent response

7:12 AM - Responded in 1m 4s

Concrete Next Steps

To ensure a smooth rollout of the updated `fct\_orders` model, the following next steps are outlined:

1. Update and Test the Model (Owner: Data Engineer, Timing: 2 hours)
Update the `fct\_orders` model to reflect the new schema changes.
Test the updated model in a staging environment to ensure correctness and performance.
2. Review and Validate Changes (Owner: Data Engineer, Timing: 1 hour)
Review the changes made to the model to ensure they align with the new schema.
Validate that the changes do not introduce any new syntax errors or inconsistencies.
3. Deploy to Staging and Validate (Owner: Data Engineer, Timing: 2 hours)
Deploy the updated model to a staging environment.
Validate that the model works as expected and produces correct results.
4. Create a Rollback Plan (Owner: Data Engineer, Timing: 1 hour)
Develop a rollback plan in case the update does not work as expected.
Identify key steps to revert the changes and restore the previous state.
5. Schedule Deployment to Production (Owner: Data Engineer, Timing: 1 hour)
Schedule a deployment window to update the `fct\_orders` model in production.
Coordinate with stakeholders to ensure minimal disruption to business operations.
6. Deploy to Production and Monitor (Owner: Data Engineer, Timing: Ongoing)
Deploy the updated model to production during the scheduled window.
Monitor the data pipeline for any issues or errors, and be prepared to execute the rollback plan if necessary.
7. Post-Deployment Review (Owner: Data Engineer, Timing: 1 hour)
Conduct a review of the deployment to identify any lessons learned or areas for improvement.
Document the changes and the rollback plan for future reference.

Owners and Responsibilities

Data Engineer: Responsible for updating and testing the model, reviewing and validating changes, deploying to staging and production, and monitoring the data pipeline.

Data Analyst: Responsible for validating the correctness of the data and ensuring that the updated model meets business requirements.

Stakeholders: Responsible for providing input and feedback on the changes and being aware of the deployment schedule and potential impact on business operations.

Timing

Update and testing: 2 hours

Review and validation: 1 hour

Deployment to staging: 2 hours

Deployment to production: 1 hour (during scheduled window)

Monitoring and rollback: Ongoing

Post-deployment review: 1 hour

By following these concrete next steps, the updated `fct\_orders` model can be safely deployed to production, minimizing downtime and ensuring data integrity.

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 27s

The one thing I'd want a second pair of eyes on is the updated SQL code in the `fct\_orders` model, specifically the changes made to access the `REGION\_CODE` column directly from the `orders` table.

It's crucial to have another person review the changes to ensure that:

1. The syntax is correct and consistent with the existing codebase.
2. The changes accurately reflect the new schema and do not introduce any unintended consequences.
3. The updated code does not break any existing dependencies or downstream processes.

A second pair of eyes can help catch any potential issues, such as:

Typos or syntax errors

Inconsistent naming conventions or coding standards

Unintended changes to the data transformation or loading process

Potential performance impacts or optimization opportunities

By having another person review the updated code, you can increase confidence in the changes and reduce the risk of errors or issues downstream.

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.