



Vantage RuntimeAI

Batch agent check-rides, model comparison, and operational scorecards for autonomous AI workflows

8.0/10

Overall score

Run summary

Scenario: dbt Test Failure

Model: meta-llama/llama-4-maverick

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

Duration: 32s

Actual run cost: \$0.002320

Projected execution cost: \$2.04/mo (40/biz day × 22 days)

Report date: July 14, 2026 - 2:49 AM EST

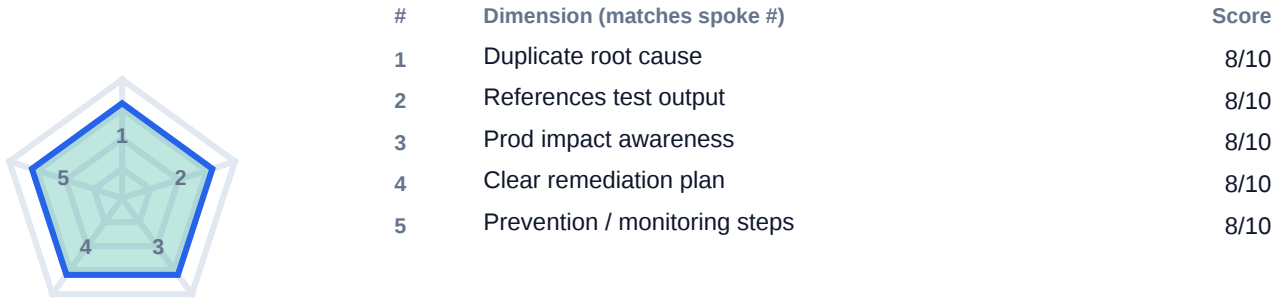
Overall

8.0/10

PASS - trust=high

Pass - score 8.0/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



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.

- Duplicate root cause identified: Met
- Immediate fix proposed: Met
- Recurrence prevention: Met
- Cites test log output: 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: 4 - Counterpart msgs: 5 - Output: 10337 chars - Latency: 32s - Criteria: 4/4

Score analysis

Calls out duplicate order_id on fct_orders from unique test failure.

Score snapshot

Solid: Solid technical task response with a few gaps.

Interpretation

Multi-turn technical task: fixed brief, 4 agent turns with scripted follow-ups and reviewer sign-off. 4 of 4 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

- + Duplicate root cause: Correctly identifies the primary technical issue from the artifacts.
- + References test output: Uses tables, logs, or schema facts from the task brief.
- + Prod impact awareness: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Clear remediation plan: Clear structure with readable SQL, diff, or memo formatting.
- + Prevention / monitoring steps: Strong rationale and actionable follow-through.

DIMENSION NOTES

- Duplicate root cause: Correctly identifies the primary technical issue from the artifacts.
- References test output: Uses tables, logs, or schema facts from the task brief.
- Prod impact awareness: Shows appropriate safety, budget, or risk awareness for the scenario.
- Clear remediation plan: Clear structure with readable SQL, diff, or memo formatting.
- Prevention / monitoring steps: 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

Duplicate root cause identified: yes - Immediate fix proposed: yes - Recurrence prevention: yes - Cites test log output: yes

CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

Transcript

Session

Scenario: dbt Test Failure

Agent response <-> Engineering counterpart

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

Duration: 32s

Engineering counterpart

6:48 AM

Task brief - dbt test failure (moderate complexity)

A uniqueness test failed in production. Triage the log and propose remediation:

...

16:02:11 Running with dbt=1.7.14

16:02:12 Found 42 models, 18 tests, 0 snapshots

16:02:12

16:02:12 Concurrency: 4 threads (target='prod')

16:02:12

16:02:12 1 of 1 START test unique_fct_orders_order_id [RUN]

16:02:14 1 of 1 FAIL 847 unique_fct_orders_order_id [FAIL 847 in 2.11s]

16:02:14

16:02:14 Finished running 1 test in 0 hours 0 minutes and 2.31 seconds (2.31s).

16:02:14

16:02:14 Completed with 1 error and 0 warnings:

16:02:14

16:02:14 Failure in test unique_fct_orders_order_id (models/marts/schema.yml)

16:02:14 Got 847 results, expected 0.

16:02:14

16:02:14 compiled Code at target/compiled/analytics/models/marts/schema.yml/unique_fct_orders_order_id.sql

16:02:14

16:02:14 select order_id, count() as n

16:02:14 from analytics.fct_orders

16:02:14 group by 1

16:02:14 having count() > 1

16:02:14

16:02:14 Sample duplicate order_id values: ORD-88421, ORD-90102, ORD-90102

...

Agent response

6:49 AM - Responded in 5s

Root Cause of Duplicates

The uniqueness test `unique\_fct\_orders\_order\_id` failed because there are duplicate `order\_id` values in the `fct\_orders` table. Specifically, the test output shows that there are 847 duplicate `order\_id` values. The sample duplicate `order\_id` values provided are `ORD-88421` and `ORD-90102`, with `ORD-90102` appearing twice in the sample.

Upon closer inspection of the compiled test code, it is evident that the duplicates are detected using the following query:

```
``sql
select order_id, count() as n
from analytics.fct_orders
group by 1
having count() > 1
``
```

This query groups the `fct\_orders` table by `order\_id` and counts the number of rows for each group. The `having count() > 1` clause filters the results to include only the `order\_id` values that appear more than once.

Immediate Fix Approach

To immediately fix the issue, we need to investigate the source of the duplicates and remove or correct them. Here's a step-by-step approach:

1. Investigate the duplicates: Run the compiled test query to retrieve a comprehensive list of duplicate `order\_id` values and their respective counts.
2. Analyze the data pipeline: Review the dbt models and data sources that feed into the `fct\_orders` table to identify potential causes of the duplicates.
3. Identify the root cause: Based on the analysis, determine the root cause of the duplicates, such as data quality issues in the source data, incorrect data transformation logic, or issues with data loading processes.
4. Correct the data: Once the root cause is identified, correct the data by either updating the source data, modifying the data transformation logic, or fixing the data loading process.

To temporarily mitigate the issue, we can consider adding a `distinct` or `row\_number()` clause to the `fct\_orders` model to remove duplicates. However, this should be done with caution, as it may mask the underlying issue.

Preventing Recurrence

To prevent similar issues from recurring in the future, we can implement the following measures:

1. Enhance data quality checks: Add more robust data quality checks to detect potential issues earlier in the data pipeline.
2. Implement data validation: Validate data at each stage of the pipeline to ensure that it conforms to expected formats and constraints.
3. Improve data transformation logic: Review and refine data transformation logic to handle edge cases and prevent duplicates.
4. Monitor data pipeline: Regularly monitor the data pipeline for anomalies and errors.
5. Add additional tests: Include additional tests, such as data quality and integrity tests, to catch potential issues before they propagate to downstream tables.

By implementing these measures, we can improve the overall reliability and quality of the data pipeline and reduce the likelihood of similar issues occurring in the future.

Engineering counterpart
6:49 AM - Responded in 0s

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

Agent response

6:49 AM - Responded in 18s

Step-by-Step Reasoning

1. Understanding the task: The task involves triaging a failed dbt test in production. The test that failed is a uniqueness test on the `order\_id` column in the `fct\_orders` table.
2. Analyzing the log output: The log output shows that the uniqueness test failed with 847 duplicate `order\_id` values. The test output includes a sample of the duplicate `order\_id` values: `ORD-88421` and `ORD-90102`, with `ORD-90102` appearing twice in the sample.
3. Interpreting the test query: The compiled test code is a SQL query that groups the `fct\_orders` table by `order\_id` and counts the number of rows for each group. The `having count() > 1` clause filters the results to include only the `order\_id` values that appear more than once.
4. Identifying the root cause: Based on the test output, it is clear that there are duplicate `order\_id` values in the `fct\_orders` table. The root cause of the duplicates is not immediately apparent from the log output alone.

Assumption: The duplicates are not a result of the test itself, but rather a genuine issue with the data in the `fct\_orders` table.

5. Proposing an immediate fix: To immediately fix the issue, we need to investigate the source of the duplicates and remove or correct them.

Assumption: The duplicates are not a result of a transient issue, but rather a persistent problem that needs to be addressed.

6. Investigating the duplicates: To investigate the duplicates, we need to run the compiled test query to retrieve a comprehensive list of duplicate `order\_id` values and their respective counts.

Assumption: The sample duplicate `order\_id` values provided in the log output are representative of the larger issue.

7. Preventing recurrence: To prevent similar issues from recurring in the future, we need to enhance data quality checks, implement data validation, improve data transformation logic, monitor the data pipeline, and add additional tests.

Assumption: The root cause of the duplicates is related to the data pipeline or data transformation logic, rather than an isolated incident.

Assumptions Made

1. The test output is accurate: The log output accurately reflects the results of the uniqueness test, and the sample duplicate `order\_id` values are representative of the larger issue.
2. The duplicates are not a result of the test itself: The duplicates are a genuine issue with the data in the `fct\_orders` table, rather than a result of the test or dbt configuration.
3. The root cause is related to the data pipeline: The duplicates are related to the data pipeline or data transformation logic, rather than an isolated incident.
4. The data pipeline is not overly complex: The data pipeline is sufficiently well-understood to allow for effective investigation and correction of the duplicates.

By making these assumptions, we can develop a reasonable plan to investigate and address the root cause of the duplicates, and implement measures to prevent similar issues from recurring in the future.

Engineering counterpart
6:49 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response
6:49 AM - Responded in 4s

Verification Steps Before Shipping to Production

Before shipping the fix to production, I would verify the following:

1. Root cause analysis is accurate: Confirm that the root cause analysis is correct, and that the duplicates are not a result of a more complex issue.
2. Fix is correctly implemented: Verify that the fix is correctly implemented, and that it addresses the root cause of the duplicates.
3. Data quality checks are in place: Ensure that data quality checks are in place to detect similar issues in the future.
4. Test coverage is adequate: Verify that test coverage is adequate to catch similar issues before they propagate to production.

Specific Verification Steps

1. Run the compiled test query: Run the compiled test query to retrieve a comprehensive list of duplicate `order\_id` values and their respective counts.
2. Verify data pipeline changes: If changes were made to the data pipeline, verify that they are correct and do not introduce new issues.
3. Run dbt tests: Run dbt tests, including the uniqueness test, to ensure that the fix resolves the issue and does not introduce new failures.
4. Verify data integrity: Verify that the data in the `fct\_orders` table is accurate and consistent after applying the fix.
5. Monitor data pipeline: Monitor the data pipeline for a reasonable period to ensure that the fix is effective and does not introduce new issues.

Production Readiness Checklist

1. Code review: Perform a code review to ensure that the changes are correct, follow dbt best practices, and are properly documented.
2. Testing: Run comprehensive tests, including dbt tests and data quality checks, to ensure that the fix is effective and does not introduce new issues.
3. Data validation: Validate data in the `fct\_orders` table to ensure that it is accurate and consistent after applying the fix.
4. Monitoring: Plan for ongoing monitoring of the data pipeline to detect potential issues early.

By verifying these items, we can ensure that the fix is effective, and that the data pipeline is reliable and accurate.

Engineering counterpart
6:49 AM - Responded in 0s

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

Agent response

6:49 AM - Responded in 4s

Top Risks

1. Inadequate root cause analysis: If the root cause analysis is incomplete or inaccurate, the fix may not address the underlying issue, and the duplicates may persist.
2. Insufficient testing: If testing is inadequate, the fix may not be thoroughly validated, and new issues may be introduced.
3. Data pipeline complexity: If the data pipeline is complex, changes may have unintended consequences, and the fix may not be effective.
4. Data quality issues: If there are underlying data quality issues, the fix may not be sufficient to resolve the problem, and additional issues may arise.
5. Downtime or data loss: If the fix is not properly planned or executed, it may result in downtime or data loss, which can have significant business impacts.

Rollback Plan

1. Revert code changes: If the fix is not effective or introduces new issues, revert the code changes to the previous version.
2. Restore previous data state: If data was modified or deleted as part of the fix, restore the previous data state from backups or snapshots.
3. Re-run previous dbt runs: Re-run previous dbt runs to re-materialize the data in the `fct\_orders` table.
4. Re-enable previous data pipeline: Re-enable the previous data pipeline configuration to ensure that data continues to flow through the pipeline.

Rollback Plan Details

1. Identify rollback triggers: Establish clear triggers for rolling back the fix, such as:
 - The fix does not resolve the duplicates issue.
 - New issues are introduced as a result of the fix.
 - Data quality issues persist or worsen.
2. Document rollback procedures: Document the rollback procedures, including the steps to revert code changes, restore previous data state, and re-run previous dbt runs.
3. Test rollback plan: Test the rollback plan to ensure that it is effective and can be executed quickly in case of an issue.
4. Communicate rollback plan: Communicate the rollback plan to relevant stakeholders, including data engineers, analysts, and business users.

By having a well-planned rollback strategy, we can minimize the risk of introducing new issues or causing downtime, and quickly recover if the fix is not effective.

Engineering counterpart

6:49 AM - Responded in 0s

Thanks - duplicate root cause and fix direction are clear. Ship the immediate dedupe/uniqueness fix; schedule the longer guardrail separately. Review complete - no further response needed.