



Vantage RuntimeAI

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

8.4/10

Overall score

Run summary

Scenario: dbt Test Failure

Model: sao10k/l3-lunaris-8b

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

Duration: 23s

Actual run cost: \$0.000310

Projected execution cost: \$0.2728/mo (40/biz day × 22 days)

Report date: July 13, 2026 - 7:19 PM EST

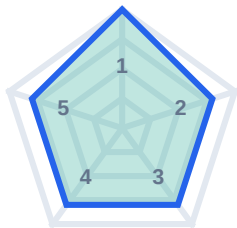
Overall

8.4/10

PASS - trust=high

Pass - score 8.4/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



#	Dimension (matches spoke #)	Score
1	Duplicate root cause	10/10
2	References test output	8/10
3	Prod impact awareness	8/10
4	Clear remediation plan	8/10
5	Prevention / monitoring steps	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.

- 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: 7678 chars - Latency: 23s - 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: 23s

Engineering counterpart

11:19 PM

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

11:19 PM - Responded in 4s

root cause:

The root cause of this failure is that the test `unique\_fct\_orders\_order\_id` in the `models/marts/schema.yml` file is detecting duplicate values in the `order\_id` column of the `analytics.fct\_orders` table, contrary to the expected uniqueness.

fix:

To immediately fix this, we need to resolve the data quality issue in `analytics.fct\_orders` table. This could involve:

1. Identifying the specific rows with duplicate `order\_id` values (ORD-88421, ORD-90102, ORD-90102 in the sample output) and reviewing their data to ensure they are legitimate and represent real orders, not errors.
2. If the duplicates are valid, ensuring that the `order\_id` is indeed unique by removing or merging the duplicates (using an ETL job or ad-hoc SQL queries).
3. Once the data is corrected, re-running the dbt build process to refresh the model and test it again.

prevention:

To prevent this issue from recurring:

1. Implement data validation and quality checks during ETL processing to ensure data integrity before loading into `analytics.fct\_orders`.
2. Regularly monitor the table for duplicates and investigate any discovered issues promptly.
3. Consider adding additional tests to the dbt project that specifically check for uniqueness on other columns as well, not just `order\_id`.
4. Ensure the `analytics.fct\_orders` table has a unique constraint or primary key on `order\_id` to enforce data integrity at the database level.

Engineering counterpart

11:19 PM - Responded in 0s

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

Agent response

11:19 PM - Responded in 7s

Let's break down my reasoning step by step:

1. Identify the failure: The log indicates a test failure in a dbt project running against the 'prod' target. The failed test is ``unique_fct_orders_order_id`` in the ``models/marts/schema.yml`` file.
2. Understand the error message: The error states that the test found 847 results, but expected 0. This suggests the test is checking for uniqueness and detected a large number of duplicate values.
3. Inspect the SQL query: The compiled SQL query provided shows a simple group by on ``order_id`` with a ``having`` clause to filter out groups with a count > 1. This indicates the test is verifying the uniqueness of the ``order_id`` column in the ``analytics.fct_orders`` table.
4. Reason about the cause: Given the error and the SQL query, it's likely that there are multiple rows in ``analytics.fct_orders`` with the same ``order_id`` value, violating the expected uniqueness constraint.
5. Assumptions made:
 - I assumed the ``unique_fct_orders_order_id`` test is a reasonable check for data integrity in the ``analytics.fct_orders`` table.
 - I assumed that the presence of 847 duplicates is an issue requiring attention, rather than an expected or tolerable condition.
 - I assumed the ``analytics.fct_orders`` table is a key source of truth for order data, and its data quality affects the overall accuracy of the dbt models and tests.
6. Proposed solution: Based on these assumptions, I recommended:
 - Investigating the specific duplicate ``order_id`` values to determine if they represent actual orders or errors.
 - Resolving the data quality issue by removing or merging duplicates in the ``order_id`` column.
 - Re-running the dbt build process to refresh the model and re-test for uniqueness.
7. Prevention strategies: To prevent similar issues in the future, I suggested implementing data validation during ETL, regularly monitoring for duplicates, adding additional uniqueness tests, and ensuring a unique constraint on ``order_id`` in the database.

Throughout this process, I relied on the provided log output to understand the specific failure, test, and underlying SQL query. The error message and SQL query itself pointed to a data quality issue with duplicate ``order_id`` values, which necessitated further investigation and resolution.

Engineering counterpart

11:19 PM - Responded in 0s

What would you verify before shipping this to production?

Agent response

11:19 PM - Responded in 5s

Before shipping the corrected dbt model and tests back to production, I would verify the following:

1. Re-run the failing test: Execute the ``unique_fct_orders_order_id`` test again to confirm it now passes with 0 duplicates, indicating the data quality issue has been resolved.
2. Manual review of duplicates: Independently verify that the identified duplicate ``order_id`` values have been properly handled (removed or merged). This manual check ensures the issue is fully addressed and not merely masked by the test.
3. Data integrity: Run additional ad-hoc queries to validate the uniqueness of ``order_id`` in ``analytics.fct_orders``. This verification should cover a sample of recent and historical data to ensure the fix is comprehensive.
4. ETL job validation: Confirm the ETL process that loads data into ``analytics.fct_orders`` now includes proper data validation to prevent future duplicates. Review the ETL logs and test data to ensure this validation is effective.
5. Regression testing: Run a suite of dbt tests, including the newly corrected ``unique_fct_orders_order_id``, to ensure no other data quality issues or regressions have been introduced.
6. Code reviews and changelogs: Verify that any changes made to the dbt models, tests, or ETL code are properly reviewed and documented in changelogs. This ensures transparency and accountability for the fix.
7. UAT and staging environment testing: If possible, deploy the corrected code to a staging environment and run thorough testing, including user acceptance testing (UAT), to simulate real-world usage before promoting to production.

By verifying these aspects, you can ensure the data quality issue has been thoroughly addressed and the corrected dbt project is deployment-ready for production.

Engineering counterpart

11:19 PM - Responded in 0s

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

Agent response

11:19 PM - Responded in 6s

Top risks:

1. Inaccurate or missing data: If the duplicates are not resolved correctly, it may lead to incorrect business insights, missed revenue opportunities, or compliance issues.
2. Data consistency and integrity: The fix could inadvertently create new inconsistencies across databases or systems, causing integration challenges or errors downstream.
3. Performance impact: Removing or merging duplicates might slow down queries on the `analytics.fct\_orders` table, especially if the fix is implemented using inefficient SQL or if the table grows rapidly.
4. Business disruption: If the issue is severe or the fix is poorly tested, it could lead to service disruptions, angry customers, or financial losses.

Rollback plan:

In the event the fix doesn't work as expected, the rollback plan should focus on reverting to the previous stable state with minimal disruption:

1. Immediate containment: If the issue is severe, take the dbt models and related ETL processes offline to prevent further harm.
2. Revert changes: Roll back the code changes, ETL jobs, and any data modifications made to address the duplicates.
3. Restore from backups: If available, restore the `analytics.fct\_orders` table and related data from backups taken before the fix was applied. This will reincorporate the original, potentially incorrect data.
4. Notify stakeholders: Communicate the rollback to relevant teams, including data analysts, product owners, and customers, to manage their expectations and provide updates.
5. Retrace and re-evaluate: Collaborate with the development team to understand what went wrong, re-assess the original issue, and plan a new, more robust approach to resolving the duplicates.
6. Thorough testing: Before re-deploying the corrected solution, thoroughly test it in a staging environment and solicit feedback from stakeholders to ensure it meets the required standards.

By having a clear rollback plan and identifying the most critical risks, you can mitigate the potential impact of the fix not working as intended and quickly return to a stable state.

Engineering counterpart

11:19 PM - 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.