



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: qwen/qwen-2.5-7b-instruct

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

Duration: 24s

Actual run cost: \$0.000880

Projected execution cost: \$0.7744/mo (40/biz day × 22 days)

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

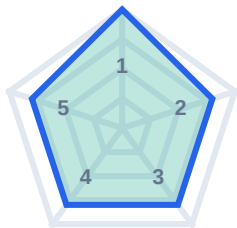
Overall

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: 11698 chars - Latency: 24s - 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: 24s

Engineering counterpart

11:18 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

11:18 PM - Responded in 5s

Root Cause Analysis

Based on the test output, the uniqueness test for the `fct\_orders` model failed because there are duplicates found in the `order\_id` field. Specifically, the test returned 847 rows where the `order\_id` is duplicated, which is unexpected since this test is designed to ensure each `order\_id` is unique.

Immediate Fix Approach

1. Identify and Correct Duplicates:

- Run a query to identify and correct the duplicates in the `fct\_orders` table.
- If the `order\_id` is supposed to remain unique, consider updating or deleting duplicate records.
- Ensure the current logic is correctly enforcing unique `order\_id` constraints in your data pipeline.

2. Re-run the Test:

- After correcting the duplicates, re-run the `unique\_fct\_orders\_order\_id` test to confirm the issue is resolved.

Example Queries

To identify duplicates in the `fct\_orders` table:

```
```sql
WITH duplicates AS (
 SELECT order_id, COUNT() as count
 FROM analytics.fct_orders
 GROUP BY order_id
 HAVING COUNT() > 1
)
SELECT order_id, count
FROM duplicates;
```
```

To correct duplicates (assuming you want to keep the first occurrence and delete the rest):

```
```sql
DELETE FROM analytics.fct_orders
WHERE order_id IN (
 SELECT order_id
 FROM (
 SELECT order_id, ROW_NUMBER() OVER (PARTITION BY order_id ORDER BY id) as rn
 FROM analytics.fct_orders
) t
 WHERE rn > 1
);
```
```

Replace `id` with the actual primary key that uniquely identifies each row.

Prevent Recurrence

1. Enforce Data Integrity Constraints:

- Add unique constraints on the `order\_id` field in your database to prevent duplicates from being inserted in the first place.

- Example in PostgreSQL:

```
```sql
ALTER TABLE analytics.fct_orders
ADD CONSTRAINT unique_order_id UNIQUE (order_id);
```
```

...

2. Data Validation in ETL Processes:

- Ensure that your ETL processes validate and clean the data before loading it into the `fct\_orders` table.
- Implement checks to ensure that `order\_id` is unique during data ingestion.

3. Regular Audits:

- Schedule regular audits to validate the uniqueness of `order\_id` and other critical fields.
- Use dbt hooks or custom scripts to run checks periodically.

4. Monitoring and Alerts:

- Set up monitoring and alerts to notify the team if the uniqueness constraint is violated.
- Use dbt hooks to run tests and alert on failures.

By following these steps, you can address the current issue and prevent similar problems from recurring in the future.

Engineering counterpart

11:18 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 6s

Certainly! Let's walk through the reasoning step by step, detailing the assumptions and logic behind each step.

Step-by-Step Reasoning

Step 1: Root Cause Analysis

Log Output:

...

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

16:02:12

16:02:12 1 of 1 FAIL 847 unique_fct_orders_order_id [FAIL 847 in 2.11s]

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 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

...

Assumptions:

1. Uniqueness Constraint: The `unique\_fct\_orders\_order\_id` test is designed to ensure that each `order\_id` in the `fct\_orders` table is unique.
2. Test Failure: The test failed, indicating that there are multiple entries with the same `order\_id`, violating the uniqueness constraint.

Step 2: Identify the Exact Issue

Test Query:

```sql

select order\_id, count() as n

from analytics.fct\_orders

group by 1

having count() > 1

```

Output:

- The query returns 847 rows where `order\_id` duplicates are present.
- Sample duplicates: `ORD-88421`, `ORD-90102`.

Assumptions:

1. Duplicate Count: There are 847 duplicate `order\_id` values.
2. Sample Values: The sample values provided (like `ORD-90102`) confirm that there are indeed duplicates.

Step 3: Immediate Fix Approach

Immediate Actions:

1. Identify and Correct Duplicates:

- To correct the duplicates, you need to determine the appropriate action (e.g., keep the first occurrence and delete the rest, or update to a consistent value).

2. Update or Delete Duplicates:

- If you want to keep the first occurrence and delete the rest:

```
```sql
DELETE FROM analytics.fct_orders
WHERE order_id IN (
 SELECT order_id
 FROM (
 SELECT order_id, ROW_NUMBER() OVER (PARTITION BY order_id ORDER BY id) as rn
 FROM analytics.fct_orders
) t
 WHERE rn > 1
);
```
```

- If you want to update the duplicates to a consistent value:

```
```sql
UPDATE analytics.fct_orders
SET order_id = 'consistent_value'
WHERE order_id IN (
 SELECT order_id
 FROM (
 SELECT order_id, ROW_NUMBER() OVER (PARTITION BY order_id ORDER BY id) as rn
 FROM analytics.fct_orders
) t
 WHERE rn > 1
);
```
```

Assumptions:

1. Primary Key: The `id` column is a unique identifier for each row.
2. Consistency: You need to decide whether to keep the first occurrence or update to a consistent value.

Step 4: Re-run the Test

Engineering counterpart

11:19 PM - Responded in 0s

What would you verify before shipping this to production?

Agent response

11:19 PM - Responded in 6s

Before shipping the fix to production, it's crucial to ensure that the issue is fully resolved and that there are no unintended consequences. Here's a step-by-step approach to verify the fix:

Step-by-Step Verification

1. Run the Fix in a Staging Environment:

- Apply the fix (e.g., deleting or updating duplicates) in a staging environment before deploying to production.
- Monitor the staging environment to ensure the fix works as expected without causing any new issues.

2. Re-run the Test in Staging:

- Re-run the `unique\_fct\_orders\_order\_id` test in the staging environment.
- Ensure that the test passes without any duplicates.

3. Check for Side Effects:

- Verify that the fix did not affect other parts of the system.
- Run additional tests to confirm that other related models and tests pass.

4. Check Data Integrity:

- Validate that the data in the `fct\_orders` table is now consistent and meets the expected uniqueness criteria.
- Run additional integrity checks to ensure all `order\_id` values are unique.

5. Review Sample Data:

- Ensure that the sample duplicates (`ORD-88421`, `ORD-90102`) have been resolved.
- Confirm that the test returns no results, indicating no duplicates.

6. Monitor Post-Fix:

- Monitor the production environment for the next 24 hours to ensure no new duplicates appear and that the system functions as expected.

Example Verification Steps

1. Verify in Staging

```
```sql
-- Delete duplicates in staging
DELETE FROM analytics.fct_orders
WHERE order_id IN (
 SELECT order_id
 FROM (
 SELECT order_id, ROW_NUMBER() OVER (PARTITION BY order_id ORDER BY id) as rn
 FROM analytics.fct_orders
) t
 WHERE rn > 1
);

-- Re-run the test in staging
SELECT order_id, count() as n
FROM analytics.fct_orders
GROUP BY 1
HAVING count() > 1;
```
```

2. Ensure No Duplicates in Production

```
```sql
```

```

-- Re-run the test in production
SELECT order_id, count() as n
FROM analytics.fct_orders
GROUP BY 1
HAVING count() > 1;
'''

3. Additional Integrity Checks
```sql
-- Check for any remaining duplicates
SELECT order_id
FROM (
    SELECT order_id, count() as count
    FROM analytics.fct_orders
    GROUP BY order_id
    HAVING count() > 1
) AS duplicates;

-- Ensure all IDs are unique
SELECT order_id, count() as count
FROM analytics.fct_orders
GROUP BY order_id
HAVING count() > 1;
'''

```

Summary

1. Staging Environment:
 - Apply the fix in staging.
 - Re-run the test to ensure it passes.
 - Verify no side effects and that all expected data integrity checks pass.
2. Production Environment:
 - Monitor the fix in production.
 - Re-run the test to confirm no duplicates.

By following these steps, you can ensure that the fix is thoroughly tested and that the system is stable after the change.

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 7s

Top Risks and Rollback Plan

Top Risks

1. Data Loss or Corruption:

- Risk: If the deletion or update process is not properly managed, it could lead to data loss or corruption.
- Mitigation: Ensure you have a backup of the `fct\_orders` table before making any changes.

2. Test Failure in Production:

- Risk: The fix might not resolve the issue in production, leading to continued test failures.
- Mitigation: Thoroughly test in a staging environment before deploying to production.

3. Inconsistent Data:

- Risk: The fix might introduce inconsistencies in the data, such as incorrect order IDs or missing data.
- Mitigation: Validate the data thoroughly after the fix.

4. Performance Issues:

- Risk: The fix might cause performance degradation in the database.
- Mitigation: Monitor the performance of the database after applying the fix.

5. Other Unintended Consequences:

- Risk: The fix might have unintended side effects on other related models or processes.
- Mitigation: Run comprehensive tests and monitor for any unexpected issues.

Rollback Plan

Rollback Steps

1. Backup the Current Database:

- Action: Take a full backup of the `fct\_orders` table.
```sql  
-- Backup the fct\_orders table  
CREATE TABLE analytics.fct\_orders\_backup AS SELECT \* FROM analytics.fct\_orders;  
```

2. Revert the Fix:

- Action: Revert the changes made during the fix.
```sql  
-- Revert the deletion of duplicates  
INSERT INTO analytics.fct\_orders (order\_id, ...) -- columns needed to match  
SELECT order\_id, ... -- columns needed to match  
FROM analytics.fct\_orders\_backup  
WHERE order\_id IN (  
 SELECT order\_id  
 FROM (  
 SELECT order\_id, ROW\_NUMBER() OVER (PARTITION BY order\_id ORDER BY id) as rn  
 FROM analytics.fct\_orders\_backup  
 ) t  
 WHERE rn > 1  
);  
  
-- Drop the backup table if no further required  
DROP TABLE analytics.fct\_orders\_backup;

```

3. Restore Data Integrity:

- Action: Restore the `fct\_orders` table to ensure it meets the expected integrity constraints.

```sql

-- Ensure the table is consistent

ALTER TABLE analytics.fct\_orders ADD CONSTRAINT unique\_order\_id UNIQUE (order\_id);

```

4. Re-run Tests:

- Action: Re-run the `unique\_fct\_orders\_order\_id` test to ensure the issue is resolved.

```sql

SELECT order\_id, count() as n

FROM analytics.fct\_orders

GROUP BY 1

HAVING count() > 1;

```

5. Monitor the System:

- Action: Monitor the system for the next 24 hours to ensure everything is working as expected.
- Monitor: Check system logs, performance metrics, and data integrity.

6. Communicate with Stakeholders:

- Action: Inform stakeholders about the rollback and the steps taken to resolve the issue.
- Communication: Send an update email or Slack message detailing the rollback process and the resolution.

Rollback Script

Here is a complete rollback script for your reference:

```sql

-- Backup the fct\_orders table

CREATE TABLE analytics.fct\_orders\_backup AS SELECT \* FROM analytics.fct\_orders;

-- Revert the deletion of duplicates

INSERT INTO analytics.fct\_orders (order\_id, ...) -- columns needed to match

SELECT order\_id, ... -- columns needed to match

FROM analytics.fct\_orders\_backup

WHERE order\_id IN (

SELECT order

**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.