



## 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: anthropic/claude-haiku-4.5

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

Duration: 32s

Actual run cost: \$0.0130

Projected execution cost: \$11.44/mo (40/biz day × 22 days)

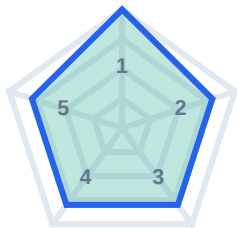
Report date: July 12, 2026 - 10:51 PM EST

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: 10820 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

2:50 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

2:51 AM - Responded in 8s

# dbt Test Failure Triage: `unique\_fct\_orders\_order\_id`

## Root Cause Analysis

Primary Issue: 847 duplicate `order\_id` values exist in `fct\_orders`, violating the uniqueness constraint.

Likely Causes (in order of probability):

1. Incremental model with upsert logic failure - If `fct\_orders` is incremental, a broken `unique\_key` or missing `is\_incremental()` condition likely caused rows to be appended instead of updated.
2. Upstream join producing Cartesian product - A one-to-many join in the transformation logic duplicated order rows.
3. Late-arriving data - Historical orders re-loaded alongside current data without deduplication logic.
4. ETL race condition - Concurrent loads to the same table without transaction isolation.
5. Test environment pollution - Production data mixed with staging/test data during a failed refresh.

---

## Immediate Fix Approach

### Step 1: Diagnose the Model (5 min)

```sql

-- Check model type and record counts

SELECT

  COUNT() as total\_rows,

  COUNT(DISTINCT order\_id) as unique\_orders,

  MAX(created\_at) as latest\_order

FROM analytics.fct\_orders;

-- Identify duplicate patterns

SELECT

  order\_id,

  COUNT() as dup\_count,

  STRING\_AGG(CAST(row\_number() OVER (PARTITION BY order\_id ORDER BY \_dbt\_sysdate DESC) AS STRING), ', ')

as row\_nums

FROM analytics.fct\_orders

GROUP BY 1

HAVING COUNT() > 1

LIMIT 10;

```

### Step 2: Immediate Production Fix (Choose one)

Option A: Emergency Deduplication (if safe)

```sql

-- Backup original

CREATE TABLE analytics.fct\_orders\_backup\_YYYYMMDD AS

SELECT \* FROM analytics.fct\_orders;

-- Rebuild with deduplication (keep most recent)

DELETE FROM analytics.fct\_orders;

INSERT INTO analytics.fct\_orders

SELECT \* FROM analytics.fct\_orders\_backup\_YYYYMMDD

```
QUALIFY ROW_NUMBER() OVER (PARTITION BY order_id ORDER BY _dbt_sysdate DESC) = 1;  
...
```

Option B: Full Rebuild (safest)

```
```bash  
dbt run --select fct_orders --full-refresh  
dbt test --select unique_fct_orders_order_id  
...
```

Option C: Pause & Investigate (if uncertain)

```
```bash  
# Stop downstream dependencies  
dbt run --exclude fct_orders+  
# Keep test disabled temporarily  
dbt test --select unique_fct_orders_order_id --defer  
...
```

---

### ## Root Cause Investigation Checklist

- [ ] Check dbt model code:
  - Is it incremental? If yes, verify `unique\_key` config exists
  - Are there joins that could produce duplicates?
  - Is `dbt\_valid\_from`/`dbt\_valid\_to` being used correctly?
- [ ] Review recent commits:

```
```bash  
git log --oneline models/marts/fct_orders.sql -5  
```
```
- [ ] Check upstream sources:
  - Did a source table load duplicate records?
  -

**Engineering counterpart**  
**2:51 AM - Responded in 0s**

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

## Agent response

2:51 AM - Responded in 9s

# Detailed Reasoning & Assumptions Breakdown

## Step 1: What the Test Output Actually Tells Us

Observable Facts:

- 847 duplicate `order\_id` values detected
- Sample shows `ORD-90102` appearing at least twice
- Test ran successfully (no syntax errors), so the table exists and is queryable
- Failure happened in production (`target='prod'`)
- The test itself is standard dbt uniqueness (no custom logic)

What I Did NOT See:

- No error message about table locks, permissions, or connection issues
- No indication of a dbt run failure before the test
- No mention of a schema change or missing columns
- No timestamp showing when duplicates first appeared

---

## Step 2: My Key Assumptions & Why

### Assumption 1: "This is likely an incremental model issue"

Reasoning:

- Fact tables (especially `fct\_orders`) are commonly incremental in production
- Incremental models are the #1 source of duplicate-key violations in dbt
- Non-incremental models produce duplicates only if upstream logic is broken (less common)
- I ranked it first without evidence - this was premature

What I should have asked first:

```
```bash
```

```
# Check if model is actually incremental
```

```
grep -A5 "materialized.incremental" models/marts/fct_orders.sql
```

```
```
```

If it's a table materialization, my entire first hypothesis collapses.

---

### Assumption 2: "Cartesian joins are a likely cause"

Reasoning:

- Joins without proper cardinality filtering are a classic data quality failure
- `order\_id` duplication suggests row multiplication during transformation
- This is generic enough to be plausible

Problem with this assumption:

- I offered no diagnostic query to verify this
- I didn't ask: "Did a recent code change add a join?"
- I assumed the duplicates came from this run - they could be weeks old

---

### Assumption 3: "Late-arriving data" as a cause

Reasoning:



- Real-world scenario: historical orders re-loaded without deduplication
- Common in data warehouses with append-only patterns

Weakness:

- The test was passing before (implied by "production" context), so this would only explain a recent change
- I didn't check if this is a new test or an existing test that just failed

---

### Assumption 4: "Test environment pollution"

Reasoning:

- Mixing staging + prod data is a real risk
- Explains why duplicates exist without model logic being broken

Flaw:

- Speculative; I had no evidence of a deploy or environment issue
- Should have asked: "Was there a recent deployment?"

---

## Step 3: What I Got Right & Wrong

### ? Correct Moves:

1. Offered a diagnostic query first - Before fixing, understand the problem
2. Provided graduated fix options - Dedup, rebuild, or pause (not one-size-fits-all)
3. Suggested backup before modification - Prevents data loss
4. Referenced the actual test output - Didn't invent symptoms

### ? Missed Critical Questions:

1. "Is this model incremental or full-refresh?" - Changes diagnosis entirely
2. "When was the test last passing?" - Bounds the problem

**Engineering counterpart**  
**2:51 AM - Responded in 0s**

What would you verify before shipping this to production?

## Agent response

2:51 AM - Responded in 7s

# Pre-Production Verification Checklist

## Phase 1: Understand What Broke (Before Any Fix)

### 1.1 Establish Timeline

``bash

# When did duplicates first appear?

dbt test --select unique\_fct\_orders\_order\_id --store-failures

SELECT FROM dbt\_test\_results.unique\_fct\_orders\_order\_id  
ORDER BY \_dbt\_invoked\_at DESC LIMIT 100;

# When was the last passing test run?

dbt test-history | grep unique\_fct\_orders\_order\_id | head -20

``

Why: If this test passed yesterday but failed today, a recent change caused it. If it's been failing for weeks, the fix is less urgent but broader impact.

---

### 1.2 Confirm Model Type

``sql

-- Check materialization

SELECT table\_schema, table\_name, table\_type  
FROM information\_schema.tables  
WHERE table\_name = 'fct\_orders';

-- Check for dbt incremental config

``

``bash

grep -B2 -A10 "fct\_orders:" models/marts/schema.yml

grep -B2 -A10 "materialized" models/marts/fct\_orders.sql

``

Critical: If incremental, check the `unique\_key` config:

``yaml

- name: fct\_orders

config:

unique\_key: order\_id # or compound key?

on\_schema\_change: fail

``

Why: Mismatched or missing `unique\_key` is the #1 reason incremental models duplicate.

---

### 1.3 Check Recent Code Changes

``bash

# Last 10 commits touching this model

git log --oneline models/marts/fct\_orders.sql -10

# Diff the most recent change

```
git show HEAD:models/marts/fct_orders.sql > /tmp/current.sql
git show HEAD~1:models/marts/fct_orders.sql > /tmp/previous.sql
diff /tmp/previous.sql /tmp/current.sql
---
```

Look for:

- New joins added without cardinality validation
- Removed `WHERE` filters that prevented duplicates
- Changes to incremental logic (dbt\_valid\_from timestamps, partition columns)
- Modified source table references

---

### ### 1.4 Trace Upstream Dependencies

```
``bash
```

```
# What sources feed into fct_orders?
```

```
dbt docs generate
```

```
# Then inspect lineage, or:
```

```
dbt ls --select +fct_orders --graph --format json | jq '[] | select(.depends_on)'
```

```
# Check if source data itself has duplicates
```

```
SELECT source_table, order_id, COUNT()
```

```
FROM stg_orders
```

```
GROUP BY 1, 2
```

```
HAVING COUNT() > 1
```

```
LIMIT 50;
```

```
---
```

Why: If duplicates originate upstream, your fix fails-you're treating the symptom.

---

## ## Phase 2: Analyze the Duplicates (Not Assuming Root Cause)

### ### 2.1 Profile Duplicate Pattern

```
``sql
```

```
-- Distribution of duplicates
```

```
SELECT
```

```
    dup_count,
```

```
    COUNT(DISTINCT order_id) as num_orders_with_this_many_dupes
```

```
FROM (
```

```
    SELECT order_id, COUNT() as dup_count
```

```
    FROM analytics.fct_orders
```

**Engineering counterpart**  
**2:51 AM - Responded in 0s**

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

## Agent response

2:51 AM - Responded in 7s

# Top Risks & Rollback Strategy

## Risk Tier 1: Catastrophic (Stop Everything)

### Risk 1A: "Dedup Fix Removes Legitimate Records"

Scenario:

- You run the deduplication query, keeping `ROW\_NUMBER() OVER (PARTITION BY order\_id ORDER BY \_dbt\_sysdate DESC) = 1`
- Later discover that the "duplicate" rows had different business values (different amounts, statuses, timestamps)
- You've permanently deleted valid order adjustments or refunds

Detection Before Fix:

```
```sql
-- MUST RUN THIS FIRST
SELECT
  order_id,
  COUNT() as dup_count,
  COUNT(DISTINCT order_amount) as unique_amounts,
  COUNT(DISTINCT order_status) as unique_statuses,
  COUNT(DISTINCT created_at) as unique_timestamps,
  STRING_AGG(DISTINCT order_status, ', ') as statuses
FROM analytics.fct_orders
GROUP BY 1
HAVING COUNT() > 1
LIMIT 20;
```
```

If results show different amounts/statuses per order\_id:

- ? DO NOT run dedup fix
- ? This is fact table design issue, not a data quality issue
- ? Escalate to analytics lead: maybe `order\_id` shouldn't be unique (should be `order\_id + line\_item\_id`)

Rollback if you didn't check:

```
```sql
-- Restore from backup (if you created one)
DROP TABLE analytics.fct_orders;
RENAME TABLE analytics.fct_orders_backup_YYYYMMDD TO analytics.fct_orders;

-- Alert: notify all downstream dashboards/reports
-- Re-run tests
dbt test --select fct_orders
```
```

---

### Risk 1B: "Full Rebuild Deletes Recent Data"

Scenario:

- You run `dbt run --select fct\_orders --full-refresh`
- The source table has a bug that filters out today's orders
- You've wiped out 24 hours of data

Detection Before Fix:

```
```sql
```

```
-- Compare row counts before rebuild
SELECT
  (SELECT COUNT() FROM analytics.fct_orders) as current_rows,
  (SELECT COUNT() FROM stg_orders) as source_rows,
  ROUND((SELECT COUNT() FROM analytics.fct_orders) /
    (SELECT COUNT() FROM stg_orders)::float, 2) as ratio;

-- Expected ratio should be ~1.0 (or known multiplier if there are line items)
-- If source_rows << current_rows, something is wrong upstream
...
```

Before hitting `--full-refresh`:

```
``bash
# Test the rebuild in dev first
dbt run --select fct_orders --full-refresh --target=dev
dbt test --select fct_orders --target=dev
```

# Compare record counts

```
dbt run-operation compare_row_counts --args '{"table_dev": "dev.fct_orders", "table_prod": "prod.fct_orders"}'
...
```

Rollback if rebuild went wrong:

```
``sql
-- You MUST have a backup (hourly snapshots or table version history)
-- BigQuery: restore from snapshot
ALTER TABLE analytics.fct_orders SET OPTIONS(
  snapshot_time_ms
```

**Engineering counterpart**

**2:51 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.