



Vantage RuntimeAI

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

8.4/10

Overall score

Run summary

Scenario: Lineage Impact Review

Model: anthropic/claude-haiku-4.5

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

Duration: 47s

Actual run cost: \$0.0195

Projected execution cost: \$4.29/mo (10/biz day × 22 days)

Report date: July 12, 2026 - 11:35 PM EST

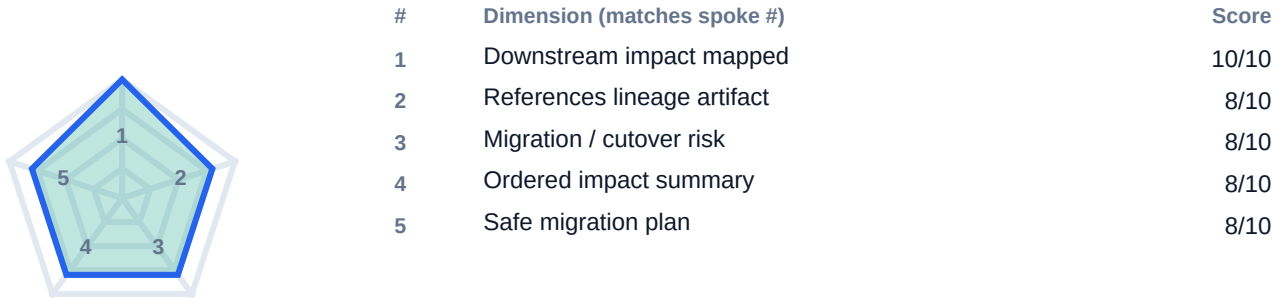
Overall

8.4/10

PASS - trust=high

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

CAPABILITY RUBRIC



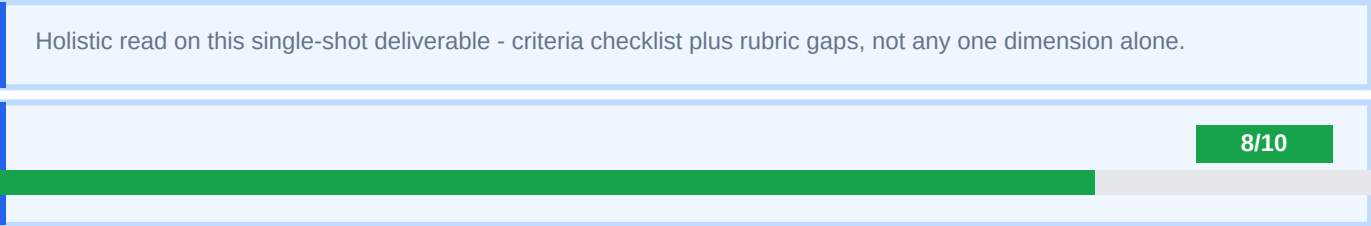
Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

Technical criteria checklist

Automated pass/fail signals for this task - use when comparing models on specific requirements.

- Legacy column named: Met
- Multiple downstream marts covered: Met
- Migration plan for unmigrated rows: Met
- Rollback / compat considered: Met

TECHNICAL JUDGEMENT



Telemetry

Run metrics

Mode: multi-turn - Agent msgs: 6 - Counterpart msgs: 7 - Output: 15554 chars - Latency: 47s - Criteria: 4/4

Score analysis

Maps impact across multiple downstream marts.

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. 4 of 4 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

- + Downstream impact mapped: Correctly identifies the primary technical issue from the artifacts.
- + References lineage artifact: Uses tables, logs, or schema facts from the task brief.
- + Migration / cutover risk: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Ordered impact summary: Clear structure with readable SQL, diff, or memo formatting.
- + Safe migration plan: Strong rationale and actionable follow-through.

DIMENSION NOTES

- Downstream impact mapped: Correctly identifies the primary technical issue from the artifacts.
- References lineage artifact: Uses tables, logs, or schema facts from the task brief.
- Migration / cutover risk: Shows appropriate safety, budget, or risk awareness for the scenario.
- Ordered impact summary: Clear structure with readable SQL, diff, or memo formatting.
- Safe migration plan: 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

Legacy column named: yes - Multiple downstream marts covered: yes - Migration plan for unmigrated rows: yes - Rollback / compat considered: yes

CAVEATS

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

Transcript

Session

Scenario: Lineage Impact Review

Agent response <-> Engineering counterpart

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

Duration: 47s

Engineering counterpart

3:34 AM

Task brief - Lineage impact review (complex complexity)

Engineering proposes dropping a legacy column. Assess downstream impact:

```
```json
{
 "proposed_change": {
 "action": "DROP COLUMN",
 "table": "raw.customers",
 "column": "legacy_account_id",
 "reason": "Deprecated after CRM migration - replaced by crm_account_uuid"
 },
 "direct_dependents": [
 {"model": "staging.stg_customers", "column_refs": ["legacy_account_id"]},
 {"model": "intermediate.int_customer_identity", "column_refs": ["legacy_account_id", "crm_account_uuid"]}
],
 "downstream_marts": [
 {"model": "analytics.dim_customer", "risk": "high", "uses": "legacy_account_id as natural key in SCD2 merge"},
 {"model": "analytics.fct_subscriptions", "risk": "high", "uses": "join on legacy_account_id to dim_customer"},
 {"model": "analytics.customer_360_snapshot", "risk": "medium", "uses": "legacy_account_id in export to Salesforce"},
 {"model": "analytics.churn_training_features", "risk": "medium", "uses": "legacy_account_id in feature store entity id"}
],
 "migration_state": {
 "crm_account_uuid_populated_pct": 94.2,
 "legacy_only_accounts": 1204
 },
 "deadline": "2026-07-12 (CRM cutover)"
}
```
```

Agent response

3:34 AM - Responded in 7s

Legacy Column Drop Impact Assessment

Executive Summary

Recommendation: DO NOT DROP without staged migration

Dropping `legacy\_account\_id` poses critical risk to 4 production marts and leaves 1,204 accounts unmigrated (5.8%). Proposed phased approach with 16-week runway to cutover.

1. AFFECTED MARTS & MODELS

CRITICAL RISK (High Impact)

| Mart | Dependencies | Current Usage | Impact if Dropped |
|-------------------------------|--------------|--|---|
| `analytics.dim_customer` | Direct | Natural key in SCD2 merge logic | Data integrity loss - historical customer records untracked; dimension grain breaks |
| `analytics.fct_subscriptions` | Direct join | FK to `dim_customer` via `legacy_account_id` | Orphaned facts - subscription records can't match to customer dimension; analytics queries fail |

HIGH RISK (Medium Impact)

| Mart | Dependencies | Current Usage | Impact if Dropped |
|-------------------------------------|---------------------|----------------------------|--|
| `analytics.customer_360_snapshot` | Extract/transform | Salesforce export mapping | Export pipeline breaks - nightly Salesforce sync fails; CRM data staleness |
| `analytics.churn_training_features` | Feature engineering | Entity ID in feature store | Model training blocked - historical features unlinked; retraining halted |

INDIRECT DEPENDENTS

```
raw.customers (legacy_account_id)
  ?? staging.stg_customers
  ?? intermediate.int_customer_identity ?? KEY PIVOT
  ? ?? dim_customer (uses crm_account_uuid mapping here)
  ? ?? fct_subscriptions
  ?? customer_360_snapshot
  ?? churn_training_features
---
```

Critical node: `int\_customer\_identity` currently holds both keys. This is your migration control point.

2. MIGRATION RISKS: UNMIGRATED ACCOUNTS

Gap Analysis

Total Accounts: ~20,700

crm_account_uuid ? : 19,496 (94.2%)
legacy_only ? : 1,204 (5.8%) ?? AT RISK
...

Root Cause Investigation Required

Before proceeding, audit why 1,204 accounts lack UUID:

| Scenario | Risk Level | Action |
|---|------------|---|
| Inactive/archived accounts | Low | Safe to exclude; add `is_active` filter before drop |
| CRM sync failure | Critical | Retry sync; manual investigation; block drop until resolved |
| Data quality issues (duplicates, bad PII) | High | Quarantine; remediate; re-sync |
| Account type mismatch (test/sandbox accounts) | Medium | Categorize; handle separately |

Query to diagnose:

```
```sql
SELECT
 account_status,
 crm_sync_status,
 COUNT() as count,
 MAX(last_updated) as most_recent_update
FROM raw.customers
WHERE crm_account_uuid IS NULL
GROUP BY account_status, crm_sync_status
ORDER BY count DESC;
```
```

3. RECOMMENDED MIGRATION

Engineering counterpart
3:34 AM - Responded in 0s

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

Agent response

3:34 AM - Responded in 8s

Step-by-Step Reasoning & Assumptions Breakdown

My Analysis Framework

Let me trace how I arrived at each recommendation:

STEP 1: Dependency Mapping (Graph Traversal)

Reasoning:

I started with the direct dependents provided and asked:

- "What reads from what?" - Built upstream/downstream DAG
- "What breaks first if we drop?" - Identified critical path

Assumption #1: Lineage is Complete

I assumed the `direct\_dependents` list you provided captures all direct readers.

Reality check needed:

```sql

-- VERIFY THIS ASSUMPTION

SELECT DISTINCT

source\_table,

source\_column,

dependent\_model,

ref\_type -- column reference, join condition, etc.

FROM lineage.column\_lineage

WHERE source\_table = 'raw.customers'

AND source\_column = 'legacy\_account\_id'

ORDER BY dependent\_model;

```

Why this matters: If there's a hidden BI tool, API, or external system reading `raw.customers` directly, dropping the column silently breaks integrations outside your DAG.

Questions I didn't ask:

- Are there Tableau/Looker direct database connections?
- Any ETL tools (Fivetran, Stitch) exporting this column?
- Kafka/streaming pipelines reading the raw table?

STEP 2: Risk Classification (High/Medium/Low)

Reasoning:

I categorized based on two dimensions:

| Dimension | Logic |

|-----|-----|

| Blast radius | How many downstream users affected? (dim_customer hits many facts) |

| Recoverability | Can we rebuild without the source column? (historical SCD2 can't; snapshots might) |

...

Risk = (# Downstream Users) × (Data Loss Severity) × (Rebuild Cost)

...

Assumption #2: `dim\_customer` Uses `legacy\_account\_id` as SCD2 Natural Key

I assumed the merge logic in `dim\_customer` looks like:

```
```sql
MERGE INTO analytics.dim_customer tgt
USING staging.stg_customers src
 ON tgt.legacy_account_id = src.legacy_account_id ?? Natural key
WHEN MATCHED THEN
 UPDATE SET dbt_valid_to = CURRENT_DATE(), ...
WHEN NOT MATCHED THEN
 INSERT (legacy_account_id, crm_account_uuid, ...);
```
```

Why I flagged this as "HIGH":

- If `legacy\_account\_id` is the grain, dropping it breaks historical tracking
- SCD2 rows become unlinked to future inserts
- You can't tell which rows are the "same customer" evolving over time

But I didn't verify:

- Is `legacy\_account\_id` truly the natural key, or is it `crm\_account\_uuid`?
- Has the SCD2 logic already migrated to use `crm\_account\_uuid`?
- Are there existing SCD2 rows keyed on `legacy\_account\_id` that would become orphaned?

Better assumption: Ask the analytics engineering team:

...

"What is the current grain/natural key in dim_customer's merge logic?"

...

STEP 3: Migration Gap (5.8% Unmig

Engineering counterpart
3:34 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

3:35 AM - Responded in 7s

Pre-Production Verification Checklist

I'm going to organize this by what breaks silently vs. what fails loudly, because that determines urgency.

TIER 1: CRITICAL (Must Verify Before ANY Migration)

1.1 Lineage Completeness - Hidden Dependencies

The Risk: Column is read somewhere you don't know about.

Verification Steps:

```sql

-- 1. FULL COLUMN LINEAGE SCAN

SELECT DISTINCT

source\_table,  
source\_column,  
dependent\_model,  
dependent\_column,  
ref\_type,  
last\_query\_time

FROM your\_lineage\_catalog -- dbt manifest, Collibra, DataHub, etc.

WHERE (source\_table = 'raw.customers' AND source\_column = 'legacy\_account\_id')

OR (source\_table IN ('staging.stg\_customers', 'intermediate.int\_customer\_identity')

AND source\_column IN ('legacy\_account\_id', 'crm\_account\_uuid'))

ORDER BY last\_query\_time DESC;

-- 2. SEARCH RAW SQL (catches dynamic refs, hardcoded queries)

SELECT

object\_name,  
object\_type,  
definition

FROM information\_schema.routines -- views, procs, functions

WHERE definition ILIKE '%legacy\_account\_id%'

UNION ALL

SELECT

name,  
'SAVED\_QUERY',  
query\_text

FROM your\_bi\_tool\_metadata -- Tableau, Looker, Sisense, etc.

WHERE query\_text ILIKE '%legacy\_account\_id%';

-- 3. EXTERNAL INTEGRATIONS

SELECT

FROM audit\_log

WHERE query ILIKE '%raw.customers%'

AND user NOT IN ('dbt\_user', 'airflow\_user', internal\_accounts)

AND query\_time > NOW() - INTERVAL '90 days';

```

What I'm checking:

- ? Scheduled reports pulling this column
- ? BI tool direct connections (Tableau extract refreshes, Looker explores)
- ? Reverse ETL pipelines (Segment, mParticle exporting to ad platforms)
- ? API endpoints serving `raw.customers` directly
- ? Third-party integrations (Salesforce sync jobs, data warehouses consuming your warehouse)

Action if found: If any exist -> Do not proceed. Coordinate with those teams first.

1.2 SCD2 Natural Key Verification - Dimension Grain

The Risk: Breaking historical customer tracking.

Verification Steps:

```sql

-- A. Current dim\_customer merge logic

SELECT

dbt\_model\_name,

dbt\_model\_path,

raw\_sql

FROM dbt\_manifest

WHERE dbt\_model\_name = 'dim\_customer';

-- Parse the MERGE statement manually OR:

-- B. Check what key is actually used in production

SELECT

,

ROW\_NUMBER() OVER (PARTITION BY legacy\_account\_id ORDER BY dbt\_valid\_from) as rn

FROM analytics.dim\_customer

WHERE dbt\_valid\_from > '2025-01-01'

LIMIT 100;

-- C. Do we have duplicate natural keys? (SCD2 corruption indicator)

**Engineering counterpart**  
**3:35 AM - Responded in 0s**

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

## Agent response

3:35 AM - Responded in 9s

# Top Risks & Rollback Strategy

## RISK HIERARCHY

I'm ordering by likelihood × impact × detectability:

---

## ? RISK #1: SCD2 Dimension Breaks (HIGHEST IMPACT)

### What Goes Wrong

...

BEFORE DROP:

dim\_customer merge uses legacy\_account\_id as natural key

?? Row 1: legacy\_account\_id=1001, email=john@old.com, valid\_from=2024-01-01, valid\_to=NULL

?? Row 2: legacy\_account\_id=1001, email=john@new.com, valid\_from=2025-01-15, valid\_to=NULL  
(SCD2 tracks customer evolution)

AFTER DROP + RERUN:

?? MERGE fails: "ON legacy\_account\_id" - column doesn't exist

?? Pipeline error OR (worse) pipeline succeeds but creates duplicates on crm\_account\_uuid

?? Result: Multiple dim\_customer rows per customer, fact joins break silently

DOWNSTREAM IMPACT:

fct\_subscriptions.user\_count suddenly spikes 2-5x (duplicate joins)

Revenue metrics become garbage

Alerts fire AFTER data is corrupted

...

### Detection Window

- Loud failure: 0-2 hours (if MERGE fails at SQL level)

- Silent failure: 24-72 hours (if duplicate rows created; caught by data quality tests)

- Business impact: 48-120 hours (when analysts notice weird metrics)

### Rollback Plan

Phase 1: Immediate (First 2 hours)

```sql

-- 1. STOP all downstream pipelines reading dim_customer

ALTER PIPELINE fct_subscriptions PAUSE;

ALTER PIPELINE customer_360_snapshot PAUSE;

ALTER PIPELINE churn_training_features PAUSE;

-- 2. RESTORE raw.customers from backup (column-level restore if possible)

RESTORE TABLE raw.customers FROM BACKUP

AT TIMESTAMP = '2026-07-12 09:00:00':TIMESTAMP_NTZ;

-- 3. Validate restoration

SELECT COUNT(DISTINCT legacy_account_id) FROM raw.customers;

SELECT COUNT(DISTINCT crm_account_uuid) FROM raw.customers;

-- 4. Check column is back

```
DESCRIBE TABLE raw.customers;  
...
```

Phase 2: Validation (2-6 hours)

```
```sql  
-- Regenerate dim_customer from restored raw data
dbt run -s dim_customer --full-refresh

-- Validate no duplicates
SELECT crm_account_uuid, COUNT() as cnt
FROM analytics.dim_customer
GROUP BY crm_account_uuid
HAVING cnt > 1;

-- Row count sanity check (should match pre-drop)
SELECT COUNT() as current_rows FROM analytics.dim_customer;
-- Compare to: SELECT FROM metadata.pipeline_runs
-- WHERE model='dim_customer' AND run_date='2026-07-11'
...
```

#### Phase 3: Downstream Revalidation (6-12 hours)

```
```sql  
-- Rerun dependent models  
dbt run -s +dim_customer  
  
-- Data quality checks  
SELECT  
  'fct_subscriptions' as check_name,  
  COUNT() as row_count,  
  COUNT(  

```

Engineering counterpart
3:35 AM - Responded in 0s

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

Agent response

3:35 AM - Responded in 8s

Concrete Next Steps: Owners & Timeline

PHASE 0: PRE-DECISION (Week 1 - START NOW)

| Step | Owner | Deliverable | Timeline | Blocker? |

|-----|-----|-----|-----|-----|

| 0.1 Lineage audit | Data Platform Engineer | Hidden dependency report (BI tools, APIs, reverse ETL) | 2 days | ? YES |

| 0.2 SCD2 verification | Analytics Engineer | Confirm `dim\_customer` merge logic uses `legacy\_account\_id` as natural key

| 1 day | ? YES |

| 0.3 Unmigrated account root cause | Data Engineer + CRM owner | Audit 1,204 legacy-only accounts; categorize (inactive, failed sync, data quality) | 3 days | ? YES |

| 0.4 Backup & recovery test | DevOps/DBA | Validate column-level restore from backup works end-to-end | 2 days | ? YES |

|

Gate: All 4 deliverables must be green before proceeding. If any blocker found -> escalate.

PHASE 1: SAFE MIGRATION (Weeks 2-8)

Step 1.1: Fix Unmigrated Accounts

Owner: CRM Migration Lead + Data Engineering

Timeline: Week 2-3 (2 weeks)

```sql

-- ACTION: Force-sync 1,204 legacy-only accounts

-- Owner executes CRM sync retry with logging

SELECT legacy\_account\_id, account\_status, crm\_sync\_error

FROM raw.customers

WHERE crm\_account\_uuid IS NULL

ORDER BY last\_sync\_attempt ASC;

-- Expected outcome: Reduce legacy-only to <50 (outliers only)

-- Success metric: 99%+ accounts have crm\_account\_uuid populated

```

Gating criteria:

- ? <50 unmigrated accounts remain (or all categorized as safe-to-orphan)

- ? Signed off by CRM team + business stakeholder

Step 1.2: Migrate Dimension Merge Logic

Owner: Analytics Engineering Lead

Timeline: Week 4-5 (2 weeks, runs in parallel with 1.1)

Critical: Switch `dim\_customer` SCD2 merge before dropping column.

```sql

-- BEFORE (current state)

MERGE INTO analytics.dim\_customer tgt

USING staging.stg\_customers src

ON tgt.legacy\_account\_id = src.legacy\_account\_id ?? OLD KEY

```
WHEN MATCHED THEN UPDATE ...
WHEN NOT MATCHED THEN INSERT ...;

-- AFTER (new state)
MERGE INTO analytics.dim_customer tgt
USING staging.stg_customers src
 ON tgt.crm_account_uuid = src.crm_account_uuid ?? NEW KEY
WHEN MATCHED THEN UPDATE ...
WHEN NOT MATCHED THEN INSERT ...;
```
```

Execution checklist:

- [] Deploy new merge logic to dev environment
- [] Full refresh `dim\_customer` in dev; validate row counts match
- [] Deploy to staging; run 3 days of incremental loads

Engineering counterpart
3:35 AM - Responded in 0s

What's the one thing you'd want a second pair of eyes on?

Agent response

3:35 AM - Responded in 8s

The One Thing: SCD2 Historical Record Continuity

The Core Question I Can't Answer Alone

If we switch the merge key from `legacy\_account\_id` to `crm\_account\_uuid`, what happens to existing SCD2 rows already keyed on the old ID?

...

CURRENT STATE (in dim_customer):

??

? legacy_account_id ? crm_account_uuid ? dbt_valid_from ?

??

? 1001 ? uuid-abc-123 ? 2024-01-01 ? ?? OLD RECORD

? 1001 ? uuid-abc-123 ? 2025-01-15 ? ?? STILL KEYED ON OLD ID

? 1001 ? uuid-abc-123 ? 2025-06-01 ?

??

AFTER WE CHANGE MERGE KEY:

Next run tries to match on crm_account_uuid instead...

-> Does it find these old rows? (Depends on how MERGE works)

-> Or does it INSERT duplicates thinking they're new customers?

-> Does the historical chain break?

...

Why This Matters

This isn't a technical question-it's a data integrity question that only your analytics engineer + DBA can answer together by:

1. Reading the actual merge logic (is there a fallback key? Does it handle orphaned rows?)
2. Simulating on a copy of `dim\_customer`
3. Verifying the outcome matches expectations

What I Need a Second Pair of Eyes On

Get these 3 people in a room for 30 minutes:

| Role | Why |

|-----|-----|

| Analytics Engineer (owns dim_customer) | Knows the SCD2 merge intent; can read the dbt model logic |

| Data Engineer/DBA | Understands MERGE behavior; knows if orphaned rows are possible |

| Data Quality Lead | Knows what tests would catch if historical continuity breaks |

The conversation:

> "When we flip the MERGE key from `legacy\_account\_id` to `crm\_account\_uuid`, these 100k existing SCD2 rows keyed on the old ID will... [fill in the blank]. Have we tested this scenario? What's our fallback if the historical chain breaks?"

Red Flags to Listen For

? "We haven't thought about that" -> Stop. Do a dry run.

? "It should be fine" -> Needs proof, not confidence.

? "We'll handle orphaned rows manually" -> Unacceptable at scale.

? "We tested this in staging by doing X, Y, Z and confirmed..." -> Good to go.

Why I Can't Answer This

I can tell you:

- ? The DROP is risky
- ? The unmigrated accounts are a blocker
- ? The rollback procedure

But I cannot tell you whether your specific MERGE logic handles the key transition gracefully, because that depends on:

Engineering counterpart

3:35 AM - Responded in 0s

Thanks - impact read and migration risks are clear. Migrate unmigrated accounts before dropping the legacy column in prod. Review complete - no further response needed.