



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: google/gemini-2.5-flash

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

Duration: 30s

Actual run cost: \$0.007950

Projected execution cost: \$1.75/mo (10/biz day × 22 days)

Report date: July 13, 2026 - 10:41 PM EST

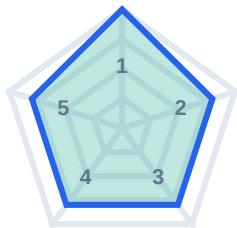
Overall

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	Downstream impact mapped	10/10
2	References lineage artifact	8/10
3	Migration / cutover risk	8/10
4	Ordered impact summary	8/10
5	Safe migration plan	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.

- Legacy column named: Met
- Multiple downstream marts covered: Met
- Migration plan for unmigrated rows: Met
- Rollback / compat considered: 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: 6 - Counterpart msgs: 7 - Output: 18918 chars - Latency: 30s - 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: 30s

Engineering counterpart

2:41 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

2:41 AM - Responded in 4s

Lineage Impact Review: Dropping `raw.customers.legacy\_account\_id`

Proposed Change: Dropping `raw.customers.legacy\_account\_id`

Reason: Deprecated after CRM migration - replaced by `crm\_account\_uuid`.

Affected Data Marts/Models:

The following data marts and models are directly or indirectly dependent on `raw.customers.legacy\_account\_id` and will be affected by its removal:

Direct Dependents (requiring immediate modification):

`staging.stg\_customers`: Directly references `legacy\_account\_id`. This staging model will need to be updated to use `crm\_account\_uuid` instead, or if `legacy\_account\_id` is only used for temporary processing, the reference removed.

`intermediate.int\_customer\_identity`: References both `legacy\_account\_id` and `crm\_account\_uuid`. This model is likely responsible for harmonizing or mapping these IDs. It will need to be updated to exclusively use `crm\_account\_uuid` and potentially handle the remaining `legacy\_only\_accounts`.

Downstream Data Marts (requiring cascading modifications):

`analytics.dim\_customer`: High Risk. Uses `legacy\_account\_id` as a natural key in a SCD2 merge. This is a critical dependency. The SCD2 merge logic must be refactored to use `crm\_account\_uuid` as the natural key. This will likely involve:

- Adding `crm\_account\_uuid` as the new natural key.

- Handling the transition for existing records where `legacy\_account\_id` was the key.

- Potentially backfilling `crm\_account\_uuid` for historical `dim\_customer` records if not already present.

`analytics.fct\_subscriptions`: High Risk. Joins on `legacy\_account\_id` to `dim\_customer`. This join condition must be updated to use `crm\_account\_uuid` from both `fct\_subscriptions` and `dim\_customer`. This likely requires `fct\_subscriptions` to also be updated to ingest `crm\_account\_uuid` from its source.

`analytics.customer\_360\_snapshot`: Medium Risk. Uses `legacy\_account\_id` in an export to Salesforce. This export process needs to be updated to use `crm\_account\_uuid` instead. This might involve coordination with the Salesforce team to ensure their ingestion process is ready for the change.

`analytics.churn\_training\_features`: Medium Risk. Uses `legacy\_account\_id` in the feature store entity ID. The feature store entity ID definition will need to be updated to use `crm\_account\_uuid`. This change requires careful consideration to avoid breaking existing models and potentially retraining models if the entity ID change impacts feature derivation or lookup.

Migration Risks for Unmigrated Accounts:

The `migration\_state` indicates:

- `crm\_account\_uuid\_populated\_pct`: 94.2%

- `legacy\_only\_accounts`: 1204

This means 1204 accounts (5.8% of the total) still rely only on `legacy\_account\_id`. Dropping `legacy\_account\_id` will make these accounts effectively invisible or unprocessable in the data warehouse, leading to:

Data Loss/Incompleteness: Data related to these 1204 accounts will not flow through the updated pipelines, affecting `dim\_customer`, `fct\_subscriptions`, `customer\_360\_snapshot`, and `churn\_training\_features`.

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

Agent response

2:41 AM - Responded in 5s

Okay, let's break down the reasoning process step-by-step, along with the assumptions I made.

Step 1: Understand the Core Request and Proposed Change

Request: Assess the impact of dropping a column.

Proposed Change: `DROP COLUMN raw.customers.legacy_account_id`.`

Reason for Change: Column is deprecated due to CRM migration, replaced by ``crm_account_uuid`.`

Initial Thought: This is a classic "find all dependencies and replace" problem. The key is to ensure the replacement (``crm_account_uuid``) is fully integrated before removing the old one.

Step 2: Analyze ``direct_dependents``

This section explicitly lists models that select directly from ``raw.customers`` and use the ``legacy_account_id``.

``staging.stg_customers`` (column_refs: ``legacy_account_id``)

Reasoning: Staging models are usually the first transformation layer. If ``legacy_account_id`` is selected here, it means subsequent models might be using it. It must be removed from this model's definition.

Assumption: ``stg_customers`` is selecting ``legacy_account_id`` from ``raw.customers``. This is explicitly stated.

Impact: The ``SELECT`` statement in ``stg_customers`` will fail.

Solution: Modify ``stg_customers`` to:

Replace ``legacy_account_id`` with ``crm_account_uuid``.

If ``legacy_account_id`` was only used for a temporary, non-persisted purpose, remove the reference entirely.

Ensure ``crm_account_uuid`` is available in ``raw.customers`` and mapped correctly.

``intermediate.int_customer_identity`` (column_refs: ``legacy_account_id``, ``crm_account_uuid``)

Reasoning: This model's name ("identity") and its reference to both IDs strongly suggest it's involved in managing the transition or mapping between old and new identifiers. It's a critical point for ensuring data consistency.

Assumption: ``int_customer_identity`` is likely performing some form of ``COALESCE``, ``CASE WHEN``, or join logic to reconcile these IDs. It probably aims to provide a unified ``customer_id`` for downstream models.

Impact: The model will break when ``legacy_account_id`` is removed.

Solution: This model needs to be updated to exclusively rely on ``crm_account_uuid``. This also highlights its importance in handling the "legacy-only accounts" problem (addressed later). It might need to carry forward the ``legacy_account_id`` for a transitional period for those specific accounts if ``crm_account_uuid`` is not available for them.

Step 3: Analyze ``downstream_marts`` (High-Risk Models First)

These are the business-facing data assets. Changes here have the highest potential for business impact.

``analytics.dim_customer`` (risk: high, uses: ``legacy_account_id`` as natural key in SCD2 merge)

Reasoning: A natural key in an SCD2 (Slowly Changing Dimension Type 2) is fundamental. It uniquely identifies a customer across changes in their attributes. If the natural key changes, the SCD2 logic must be fundamentally re-engineered.

Assumption: The ``dim_customer`` table has its own surrogate key, but uses ``legacy_account_id`` to

Engineering counterpart
2:41 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

2:41 AM - Responded in 5s

Before shipping the changes related to dropping ``raw.customers.legacy_account_id`` to production, I would perform a comprehensive verification process, encompassing technical checks, data validation, and stakeholder communication.

Here's a detailed verification plan:

1. Code Review and Best Practices:

Comprehensive Code Review:

Ensure all modified models (``stg_customers``, ``int_customer_identity``, ``dim_customer``, ``fct_subscriptions``, ``customer_360_snapshot``, ``churn_training_features``, and any others) have been reviewed by at least one other senior data engineer.

Verify adherence to coding standards, documentation, and data modeling best practices.

Check for clear comments explaining the logic, especially around ``crm_account_uuid`` adoption and ``legacy_account_id`` deprecation.

Idempotency & Re-runability: Confirm that all transformations are idempotent and can be re-run without producing inconsistent results.

Performance Optimization: Review query plans for new joins and transformations to ensure optimal performance, especially for high-volume tables like ``fct_subscriptions``.

Error Handling: Ensure robust error handling mechanisms are in place for any new logic, especially for data type mismatches or null ``crm_account_uuid`` values.

2. Data Validation & Testing:

Unit/Component Testing:

For each modified model, create dedicated tests using mocked or small, controlled datasets.

Test specific scenarios: full ``crm_account_uuid`` population, ``legacy_only_accounts``, null values, and edge cases.

Integration Testing:

Run tests that cover the entire affected pipeline, from ``raw.customers`` through to the final analytics marts.

Verify data flow and consistency across models (e.g., ``crm_account_uuid`` joining correctly from ``dim_customer`` to ``fct_subscriptions``).

Data Integrity Checks (Pre- and Post-Deployment):

Record Counts: Compare ``COUNT()`` for key tables (``dim_customer``, ``fct_subscriptions``) before and after the change on a test environment.

Key Uniqueness: Verify that ``crm_account_uuid`` is unique where expected (e.g., ``dim_customer`` natural key, ``int_customer_identity`` primary key).

Referential Integrity: For tables relying on ``crm_account_uuid`` for joins (e.g., ``fct_subscriptions`` to ``dim_customer``), check that all ``crm_account_uuid`` values in the fact table have a corresponding entry in the dimension table.

Data Completeness: Sample data to ensure no unexpected nulls or missing data in critical columns.

Value Distribution: Check ``MIN()``, ``MAX()``, ``AVG()``, ``DISTINCT COUNT()`` for key numeric and categorical columns to detect anomalies introduced by the change.

"Legacy-Only Accounts" Validation:

Pre-Deployment: Run a query to identify the exact ``legacy_only_accounts`` (e.g., ``SELECT legacy_account_id FROM raw.customers WHERE crm_account_uuid IS NULL``).

Post-Deployment (Test Env): Verify that these specific accounts are handled as per the agreed-upon strategy (e.g., assigned a placeholder ``crm_account_uuid``, filtered out, or explicitly flagged). If they are to be filtered out, verify they are indeed gone from the downstream models. If they are to be mapped, verify the mapping logic.

SCD2 Validation for

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

Agent response

2:41 AM - Responded in 6s

Okay, let's consolidate the top risks and formulate a robust rollback plan.

Top Risks Associated with Dropping `raw.customers.legacy\_account\_id`

The primary risks stem from the fundamental nature of `legacy\_account\_id` as a natural key and the existence of unmigrated accounts.

1. Data Discrepancy & Loss for "Legacy-Only Accounts" (High Risk - Most Critical)

Description: The 1204 accounts that only have `legacy\_account\_id` will effectively become "invisible" to the data warehouse if `crm\_account\_uuid` is the sole identifier. This means no data for these customers will flow into `dim\_customer`, `fct\_subscriptions`, `customer\_360\_snapshot`, or `churn\_training\_features`.

Consequences:

- Inaccurate reporting, dashboards, and analytics metrics for customer count, revenue, activity, churn, etc.

- Broken customer 360 views for these specific customers.

- Churn models potentially missing crucial data points, leading to reduced accuracy.

- Operational issues if Salesforce exports rely on these accounts.

Mitigation (Pre-emptive): This risk must be addressed before deployment. The strategy for these accounts (e.g., generating placeholder `crm\_account\_uuid`s, holding off on the drop until they're migrated, or accepting the data loss for these specific accounts) needs explicit stakeholder sign-off. My prior recommendations focused on migrating these, which is the ideal scenario.

2. Broken `dim\_customer` SCD2 Logic & Data Integrity (High Risk)

Description: Changing the natural key for an SCD2 dimension is a complex operation. If the migration to `crm\_account\_uuid` as the natural key is flawed, it can lead to:

- Incorrect `dim\_customer` history (e.g., new versions not being created, old versions not being expired).

- Duplicate customer entries or orphaned entries.

- Loss of historical attribute changes if the join/merge logic is faulty.

Consequences: Flawed historical analysis, incorrect joins to fact tables, and general data trust erosion.

Mitigation: Extremely rigorous testing on a representative dataset, including historical simulations for SCD2 merges. Extensive data validation comparing pre- and post-migration `dim\_customer` records.

3. Broken Joins in `fct\_subscriptions` & Downstream Models (High Risk)

Description: `fct\_subscriptions` uses `legacy\_account\_id` to join to `dim\_customer`. If `fct\_subscriptions` is updated to use `crm\_account\_uuid` but `dim\_customer` isn't correctly populated with `crm\_account\_uuid` for all relevant historical records, or if the join itself is incorrect.

Consequences: Missing subscription data, incorrect revenue reporting, and inability to link subscriptions to customer attributes.

Mitigation: Ensure `crm\_account\_uuid` is backfilled or universally available in `dim\_customer` before `fct\_subscriptions` is changed. Thorough integration testing of the join.

4. Impact on External Systems (Salesforce Export, Feature Store) (Medium-High Risk)

Description: `customer\_360\_snapshot` exports `legacy\_account\_id` to Salesforce, and `churn\_training\_features` uses it as an entity ID. If these external systems are not updated in sync, data exports will fail or be misinterpreted, and feature store lookups will break.

Engineering counterpart
2:41 AM - Responded in 0s

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

Agent response

2:41 AM - Responded in 5s

Okay, let's turn this into a concrete action plan with Owners and Timelines, assuming the ideal outcome where we aim for 100% `crm\_account\_uuid` coverage.

Action Plan: Legacy Column Dropping (`raw.customers.legacy\_account\_id`)

Goal: Safely drop `raw.customers.legacy\_account\_id` by the CRM cutover deadline (2026-07-12) with minimal business disruption.

Key:

- Owner (Data Eng): Data Engineering Team responsible for implementation and testing.
- Owner (Infra/DBA): Database Administration or Data Infrastructure Team.
- Owner (CRM/Source): CRM/Source System Team responsible for data completeness.
- Owner (Analytics Eng): Analytics Engineering Team (if separate from Data Eng).
- Owner (Biz Stakeholder): Business/Product Owner of affected analytics products.
- Owner (ML Eng): Machine Learning Engineering Team.

Phase 1: Pre-Migration & Readiness (Target: T-12 weeks from deadline)

| Step | Description | Owner | Timeline | Deliverable | Notes |
|------|--------------------------------|--------------------------|-------------------|--|---|
| 1.1 | Resolve "Legacy-Only Accounts" | CRM/Source / Data Eng | T-12 to T-8 weeks | 100% `crm_account_uuid_populated_pct` OR Formalized strategy for remaining accounts with Biz Stakeholder sign-off. | Immediate Priority: Escalate to CRM/Source System team. Action: Get `crm_account_uuid` populated for all 1204 `legacy_only_accounts` by pushing data to CRM or directly updating `raw.customers` via a one-off process. Decision: If not possible, formalize strategy for these accounts (e.g., placeholder UUIDs, filtering) with Biz Stakeholder. CRITICAL PATH. This determines if a smooth migration is possible. If not 100%, adjust subsequent steps for placeholder/filtering logic. |
| 1.2 | External System Coordination | Data Eng / Analytics Eng | T-12 to T-8 weeks | Documented plan for Salesforce export update & Feature Store entity ID change. | Salesforce: Coordinate with Salesforce team to ensure their ingestion process is ready to consume `crm_account_uuid` instead of `legacy_account_id` from `analytics.customer_360_snapshot` export. Feature Store: Coordinate with ML Eng to define the transition plan for `churn_training_features` entity ID (e.g., phased rollout, retraining, impact assessment). Ensure external teams are aware and ready for changes. |

Phase 2: Development & Internal Testing (Target: T-8 to T-4 weeks from deadline)

| Step | Description | Owner | Timeline | Deliverable | Notes |
|------|--------------------------------------|----------|-----------|-------------|--|
| 2.1 | Update Staging & Intermediate Models | Data Eng | T-8 to T- | | Modify `staging.stg_customers` to select `crm_account_uuid` and drop `legacy_account_id`. Modify `intermediate.int_customer_identity` to exclusively use `crm_account_uuid` (or handle placeholder if applicable). Ensure new identifier is surfaced as primary key. |

Engineering counterpart

2:41 AM - Responded in 0s

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

Agent response

2:41 AM - Responded in 4s

The one thing I would want a second pair of eyes on, and where I would prioritize external review, is:

The entire implementation of the ``analytics.dim_customer`` SCD2 logic using ``crm_account_uuid`` as the new natural key, specifically focusing on its ability to correctly handle historical data and the transition period.

Why this is the most critical:

1. Fundamental Data Integrity: ``dim_customer`` is a core dimension. If its SCD2 logic is flawed, it corrupts historical customer attributes, which impacts nearly all downstream analytics. Incorrect historical data can lead to wrong business decisions for years.
2. SCD2 Complexity: SCD2 is inherently complex, especially when changing the natural key. It's not just about updating a ``WHERE`` clause; it involves merge logic, identifying changes, creating new rows, expiring old rows, and ensuring the ``valid_from`/`valid_to`` dates are accurate.
3. Historical Backfill: The plan involves backfilling ``crm_account_uuid`` into existing ``dim_customer`` records. This is a high-risk operation that, if done incorrectly, can permanently damage the historical record.
4. Cascading Impact: ``dim_customer`` feeds ``fct_subscriptions`` and likely many other analytical models. Any error here will propagate and break downstream joins and aggregations.
5. "Legacy-Only Accounts" Strategy: How ``dim_customer`` handles the 1204 ``legacy_only_accounts`` (whether by placeholder, explicit filtering, or waiting for full migration) will be reflected most prominently here. The reviewer should ensure this strategy is correctly implemented and aligns with the business decision.

What to look for during the second review:

Correctness of the Merge Logic: Does it accurately identify new customers, updated attributes, and historical changes?

Natural Key Transition: How is the switch from ``legacy_account_id`` to ``crm_account_uuid`` managed for existing records? Are historical versions of customers still correctly identified and linked?

Backfill Logic: Is the backfill of ``crm_account_uuid`` into existing ``dim_customer`` records robust and idempotent? Does it correctly apply to all relevant historical versions?

``valid_from`/`valid_to`` Management: Are these dates correctly maintained and updated during the transition and for new records?

Edge Cases: How does it handle nulls, duplicates, or out-of-order data for ``crm_account_uuid``?

Test Coverage: Does the testing strategy for ``dim_customer`` adequately cover historical scenarios and the SCD2 logic?

A second pair of eyes on this specific component would provide invaluable assurance that the foundational customer dimension remains accurate and reliable during and after this critical migration.

Engineering counterpart

2:41 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.