



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: openai/gpt-4.1-mini

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

Duration: 1m 16s

Actual run cost: \$0.006960

Projected execution cost: \$1.53/mo (10/biz day × 22 days)

Report date: July 14, 2026 - 3:02 AM EST

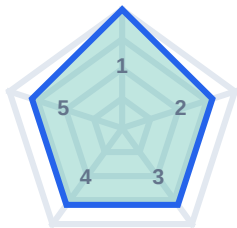
Overall

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: 18631 chars - Latency: 76s - 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: 1m 16s

Engineering counterpart

7:01 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

7:01 AM - Responded in 16s

Here is the detailed impact assessment and recommendations for dropping the `legacy\_account\_id` column from `raw.customers`:

1. Affected Marts/Models

Direct Dependents

- staging.stg_customers
 - Uses `legacy\_account\_id`
 - Will break if column dropped before migration
- intermediate.int_customer_identity
 - Uses both `legacy\_account\_id` and `crm\_account\_uuid`
 - Critical for identity resolution, needs both columns available until full migration

Downstream Marts (Analytics Layer)

- analytics.dim_customer
 - Risk: High
 - Uses `legacy\_account\_id` as natural key in SCD2 merge logic
 - Dropping the column without migrating merge logic will cause data corruption or failure
- analytics.fct_subscriptions
 - Risk: High
 - Joins to `dim\_customer` on `legacy\_account\_id`
 - Dropping column will break joins and metrics dependent on subscriptions
- analytics.customer_360_snapshot
 - Risk: Medium
 - Uses `legacy\_account\_id` for Salesforce exports
- analytics.churn_training_features
 - Risk: Medium
 - Uses `legacy\_account\_id` in feature store entity IDs

2. Migration Risks for Unmigrated Accounts

- Current State
 - `crm\_account\_uuid` populated for 94.2% of accounts
 - 1,204 accounts only have `legacy\_account\_id` (no `crm\_account\_uuid`)
- Risks
 - Dropping `legacy\_account\_id` column before migrating these accounts leads to:
 - Loss of ability to link or identify these legacy-only accounts downstream
 - Data loss for these accounts in marts relying on legacy keys
 - Broken joins and failed data refreshes in dependent models using legacy keys

3. Recommended Migration Order & Strategy

| Step | Action | Notes |

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

| 1 | Complete Migration of Legacy-Only Accounts | Prioritize population of `crm\_account\_uuid` for remaining ~5.8% legacy-only accounts before drop. |

| 2 | Update & Test Models | Refactor `staging.stg\_customers` and `intermediate.int\_customer\_identity` to fully support

`crm\_account\_uuid` and remove reliance on `legacy\_account\_id`. |

| 3 | Modify Downstream Marts | Update `dim\_customer` SCD2 logic to use `crm\_account\_uuid` as natural key; update joins in `fct\_subscriptions` and exports in `customer\_360\_snapshot` and feature store for `churn\_training\_features`. |

| 4 | Dual Key Period (Parallel Run) | For a fixed period, keep both columns and support both keys in downstream marts to validate correctness and stability. |

| 5 | Drop Column | After validation and stabilization (post 100% migration), execute DROP COLUMN on `legacy\_account\_id` from `raw.customers` and all dependent models. |

| 6 | Monitor & Optimize | Monitor data integrity and query performance post-drop. |

4. Rollback Considerations

- Before Drop:
 - Column present, rollback trivial (no immediate risk)
- After Drop:
 - Rolling back requires:
 - Re-adding `legacy\_account\_id` column to source and models
 - Backfilling data for unmigrated accounts (may require restoring backups or re-extracting from source systems)
 - Potentially complex and time-consuming
- Recommendation:
 - Avoid dropping column until all migration and model refactor complete
 - Maintain backups and run dual-key approach to minimize rollback risk

Summary

| Aspect | Status / Recommendation |
|---------------------|--|
| Legacy column usage | Critical in staging, intermediate, and multiple downstream marts |

Engineering counterpart
7:01 AM - Responded in 0s

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

Agent response

7:01 AM - Responded in 14s

Certainly! Here's a detailed walkthrough of my reasoning process, including assumptions made at each step:

Step 1: Understand the Proposed Change & Context

- Change: Drop `legacy\_account\_id` from `raw.customers`
- Context: `legacy\_account\_id` is deprecated, replaced by `crm\_account\_uuid`
- Deadline: Full cutover by 2026-07-12

Assumptions:

- The `crm\_account\_uuid` is intended as the future sole account identifier.
- The column drop is irreversible and would affect all dependent layers.
- The 2026 deadline is a hard cutover date for fully retiring legacy identifiers.

Step 2: Identify All Affected Models and Their Usage

- Direct dependents use the column directly in transformations.
- Downstream marts use it variously for keys, joins, exports, and feature IDs.
- Risk levels assigned in input data reflect business criticality and impact severity.

Assumptions:

- The listed models cover the entire direct and indirect lineage impacted by the column.
- The column references in models are accurate and complete.
- "High" risk means production break or data corruption if changed prematurely.
- "Medium" risk means some degradation, partial functionality loss, or downstream impact.

Step 3: Analyze Migration Progress and Impact

- 94.2% of accounts have `crm\_account\_uuid` populated, 5.8% remain legacy-only.
- Legacy-only accounts depend entirely on `legacy\_account\_id`.

Assumptions:

- Legacy-only accounts cannot be identified or joined without the legacy column.
- If the column is dropped now, the 5.8% of accounts will be lost downstream.
- The migration effort is ongoing but not yet complete, with a defined deadline.

Step 4: Determine Migration Risks

- Immediate drop risks breaking staging and intermediate models which directly reference `legacy\_account\_id`.
- Downstream marts rely on legacy keys for merges and joins; dropping prematurely causes failures or data integrity issues.
- Legacy-only accounts would effectively become orphaned without `legacy\_account\_id`.

Assumptions:

- Downstream models cannot be trivially refactored to use only `crm\_account\_uuid` before migration complete.
- Dual key support is possible for a transition period.
- Data refreshes and production pipelines depend heavily on these keys.

Step 5: Define Migration Order & Strategy

- Prioritize completing migration of legacy-only accounts to `crm\_account\_uuid` first.
- Refactor staging/intermediate models to remove reliance on legacy column.
- Update downstream marts to use `crm\_account\_uuid` as the primary key.
- Use a parallel run period supporting both keys to validate correctness.
- Then safely drop the legacy column.

Assumptions:

- The migration team can prioritize and complete the remaining 5.8% accounts.
- Refactoring and testing of models is feasible and can be coordinated with migration.
- Parallel-key support is viable without significant performance or complexity issues.
- The timeline allows for careful, staged rollout before 2026 deadline.

Step 6: Consider Rollback Implications

- Before drop, rollback is trivial since column is still present.
- After drop, rollback requires recreating the column and refilling data, which is complex.
- Backups and dual-key periods reduce rollback needs.

Assumptions:

- Backups and historical data snapshots exist and are reliable.
- Re-populating legacy keys after drop is a non-trivial, last-resort operation.
- Rollback is costly and disruptive-avoid post-drop except in emergencies.

Summary of Assumptions

| Assumption | Reasoning |

| - | - |

| `crm\_account\_uuid` is final replacement key | Given migration direction stated explicitly |

| Models and lineage coverage is complete | Provided data includes direct dependents and all relevant marts |

| Legacy-only accounts must be migrated before drop

Engineering counterpart
7:01 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:02 AM - Responded in 10s

Before approving and shipping the column drop to production, I would verify the following critical areas to minimize risk and ensure a smooth migration:

1. Complete Migration Verification

- Legacy-only accounts migration status:
 - Confirm that all 1,204 legacy-only accounts have been successfully and accurately assigned a `crm\_account\_uuid`.
 - Ensure data quality and uniqueness of `crm\_account\_uuid` for these accounts.
 - Verify that no new accounts have appeared without a `crm\_account\_uuid` since last check.

2. Model Refactoring and Testing

- Code changes:
 - Confirm all references to `legacy\_account\_id` are removed or replaced in direct dependents (`staging.stg\_customers`, `intermediate.int\_customer\_identity`).
 - Confirm downstream marts and transformations (`dim\_customer`, `fct\_subscriptions`, etc.) now rely solely on `crm\_account\_uuid` or dual keys as per migration plan.
- Testing:
 - Run full integration test suite, including data quality checks, to validate correctness of models without `legacy\_account\_id`.
 - Validate SCD2 merge logic in `dim\_customer` uses `crm\_account\_uuid` correctly with no regression.
 - Verify joins in `fct\_subscriptions` and exports in `customer\_360\_snapshot` produce expected results.
- Performance:
 - Measure query performance post-refactor to ensure there is no degradation from key changes.

3. Parallel Run & Dual-Key Validation

- Confirm the environment has successfully run in dual-key mode (both `legacy\_account\_id` and `crm\_account\_uuid` present) for a suitable period, demonstrating:
 - Data parity between legacy and new keys.
 - No pipeline failures or inconsistencies.
 - Downstream consumers are ready for full migration.

4. Backups and Recovery Plans

- Ensure backups of the following are recent and accessible:
 - `raw.customers` table with `legacy\_account\_id` data.
 - Full downstream data marts and models pre-change.
- Document clear rollback procedures in case the column drop causes unforeseen issues after deployment.

5. Stakeholder Sign-off

- Obtain approvals from:
 - Data consumers relying on legacy keys (analytics, data science, business teams).
 - CRM and migration teams confirming completeness of account UUID population.
 - Platform/engineering teams confirming readiness of code and infrastructure.

6. Monitoring and Alerting Setup

- Implement or verify monitoring on key metrics post-drop:
 - Data completeness and row counts in critical marts.
 - Join success rates and merge operations in key models.
 - Downstream pipeline health and latency.
- Set up alerts for anomalies or missing data related to account identifiers.

7. Communication and Scheduling

- Schedule the deployment around a low-impact window to reduce business disruption.
- Communicate timelines, expectations, and fallback plans clearly to all stakeholders.

In summary:

I would only approve shipping if I have verified complete legacy-to-UUID migration, tested and validated the updated models end-to-end, established rollback and monitoring mechanisms, and secured stakeholder sign-off. This ensures that dropping `legacy\_account\_id` will not disrupt production data pipelines or downstream analytics.

Engineering counterpart
7:02 AM - Responded in 0s

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

Agent response

7:02 AM - Responded in 12s

Certainly! Here's a focused summary of the top risks and a rollback plan if dropping `legacy\_account\_id` encounters issues in production:

Top Risks

| Risk | Impact | Mitigation |

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

- | 1. Incomplete Migration of Legacy-Only Accounts | Loss or orphaning of 1,204 accounts not mapped to `crm\_account\_uuid`, causing data gaps and broken joins downstream. | Verify and complete migration before drop; use dual-key support for fallback. |
- | 2. Downstream Models Break or Return Incorrect Data | High-risk marts like `dim\_customer` or `fct\_subscriptions` fail or produce corrupted output, affecting reporting and business decisions. | Perform thorough testing, including SCD2 logic validation and join correctness, before dropping column. |
- | 3. Pipeline Failures in Staging/Intermediate Layers | Data pipelines fail at `staging.stg\_customers` or `intermediate.int\_customer\_identity` due to missing column, halting downstream refreshes. | Coordinate code refactoring and deployment to ensure compatibility; run parallel key period. |
- | 4. Data Quality Degradation | Mismatches or duplicates in customer identifiers leading to inaccurate analytics or training features. | Monitor data quality actively post-deployment; validate outputs with historical data. |
- | 5. Insufficient Monitoring and Alerts | Issues go undetected, causing prolonged business impact. | Implement robust monitoring, alerting, and quick-response processes. |

Rollback Plan

Preconditions:

- Keep full backups of `raw.customers` table and all relevant marts/models with `legacy\_account\_id` intact before drop.
- Maintain code versions that support `legacy\_account\_id` for immediate redeployment.

Steps to Roll Back

1. Reinstate Column in Source and Models:

- Re-add `legacy\_account\_id` column in the `raw.customers` table if physically dropped (via DB restore or alter table add + backfill).
- Revert transformations and models (`staging`, `intermediate`, downstream marts) to the last known good version referencing `legacy\_account\_id`.

2. Backfill Missing Data:

- Restore data for `legacy\_account\_id` from backups or source system extracts to ensure completeness for legacy-only accounts.

3. Validate Model Outputs:

- Run tests to confirm all downstream models correctly consume `legacy\_account\_id` again.

4. Restart Data Pipelines:

- Resume production data refreshes and monitor pipelines closely.

5. Communicate Rollback:

- Inform stakeholders immediately of rollback to manage expectations and follow-up actions.

Additional Notes

- Rollback is non-trivial: If column is dropped without backups or if data for legacy accounts is lost, recovery may require costly manual re-extraction or reconciliation.
- Avoid rollback through careful planning and validation: The best risk mitigation is a cautious, staged rollout with full migration and dual-key validation.
- Document rollback steps and train the team: Ensure readiness to execute rollback rapidly if needed.

Summary:

Top risks center on incomplete migration and breakage of critical models causing data loss or pipeline failures. The rollback plan involves re-adding and backfilling the dropped column, reverting code, and restoring pipeline operations, enabled by thorough backups and version control.

Engineering counterpart
7:02 AM - Responded in 0s

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

Agent response

7:02 AM - Responded in 17s

Certainly! Below is a concrete, actionable next-step plan with clear owners and suggested timing to mitigate risks and prepare for the safe drop of `legacy\_account\_id`:

Concrete Next Steps for Dropping `legacy\_account\_id`

| Step | Action | Owner | Timing | Notes |
|-------|--|--|--------------------------------|---|
| ----- | ----- | ----- | ----- | ----- |
| 1 | Complete migration of legacy-only accounts (1,204 accounts): Validate all have `crm_account_uuid` populated and unique | CRM Migration Team | 4 weeks | High priority to eliminate legacy-only accounts before changes |
| 2 | Refactor and test staging models (`stg_customers`, `int_customer_identity`) to remove `legacy_account_id` dependency | Data Engineering | 3 weeks (parallel with Step 1) | Unit & integration tests required; deploy to DEV & QA |
| 3 | Update downstream marts to support only `crm_account_uuid` or dual keys: `dim_customer`, `fct_subscriptions`, `customer_360_snapshot`, `churn_training_features` | Analytics Engineering | 4 weeks (start after Step 2) | Include validation of SCD2 merges and joins; run data quality checks |
| 4 | Run parallel-key (dual key) period in QA and staging environments: Validate data parity and pipeline stability | Data Engineering + Analytics Engineering + QA Team | 4 weeks | Enables smooth transition and early issue detection |
| 5 | Stakeholder review & sign-off: Present results of migration completeness and testing to business, analytics, and platform teams | Data Engineering Lead + Product Owners | 1 week (after Step 4) | Obtain written approval to proceed to production drop |
| 6 | Schedule and communicate production deployment window | Data Engineering Lead + Release Management | 1 week (post sign-off) | Preferably during low-impact business period |
| 7 | Deploy DROP COLUMN in production along with updated model changes | Data Engineering | Day 0 (deployment day) | Monitor pipelines and dashboards closely post-deployment |
| 8 | Post-deployment monitoring and validation | Data Engineering + Analytics Engineering | 2 weeks post-deployment | Trigger alerts on anomalies, verify data integrity, report issues immediately |
| 9 | Prepare and document rollback procedures, ensure backups are accessible | Data Engineering + Platform Team | Before Day 0 | Rollback training session with on-call team |
| 10 | If issues detected, execute rollback plan immediately | Data Engineering + Platform Team | As needed | Restore column and data, redeploy legacy-dependent code versions |

Summary Table

| Timeline | Milestone | Owner |
|----------------------|---|--|
| ----- | ----- | ----- |
| Weeks 1-4 | Legacy-only accounts migration completion | CRM Migration Team |
| Weeks 1-4 (parallel) | Staging model refactoring & testing | Data Engineering |
| Weeks 5-8 | Downstream marts update & testing | Analytics Engineering |
| Weeks 9-12 | Parallel-key validation & QA | Cross-functional teams |
| Week 13 | Stakeholder sign-off | Data Engineering Lead & Product Owners |
| Week 14 | Schedule and communicate deployment | Data Engineering Lead & Release Mgmt |
| Week 15 (Day 0) | Production deployment & drop column | Data Engineering |
| Weeks 15-17 | Post-deployment monitoring | Data & Analytics Engineering |
| Ongoing | Rollback readiness & execution if needed | Data Engineering & Platform Team |

This plan aligns owners with clear deadlines and enforces rigorous testing and stakeholder communication before any irreversible production change.

Engineering counterpart**7:02 AM - Responded in 0s**

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

Agent response**7:02 AM - Responded in 7s**

The one critical area I'd want a second pair of eyes on is the correctness and completeness of the migration of legacy-only accounts to `crm\_account\_uuid`, specifically:

- Verification that all 1,204 legacy-only accounts are reliably populated with valid, unique `crm\_account\_uuid`s in the source system.
- Validation that no orphaned or partially migrated records remain which could cause silent data loss or join failures downstream once `legacy\_account\_id` is dropped.
- Confirmation that this migration data quality has been independently audited or reconciled against business records and stakeholder expectations.

Why?

Because dropping the legacy column prematurely without these accounts fully and cleanly migrated is the biggest risk for data loss, broken pipelines, and downstream analytical errors-impacting critical business metrics and operations.

A fresh, independent validation here reduces the chance of costly rollback or production incidents, making it the highest-value checkpoint before proceeding.

Engineering counterpart**7:02 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.