



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-4o-mini

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

Duration: 1m 5s

Actual run cost: \$0.002610

Projected execution cost: \$0.5742/mo (10/biz day × 22 days)

Report date: July 13, 2026 - 10:49 PM EST

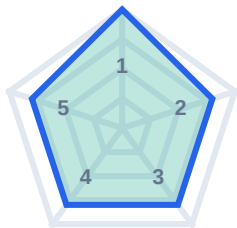
Overall

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: 19701 chars - Latency: 65s - 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 5s

Engineering counterpart

2:48 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

2:48 AM - Responded in 12s

Affected Marts/Models

1. Staging Model

- Model: `staging.stg\_customers`
 - Impact: Direct reference to `legacy\_account\_id`. Dropping this column will break transformations dependent on it.

2. Intermediate Model

- Model: `intermediate.int\_customer\_identity`
 - Impact: Uses both `legacy\_account\_id` and `crm\_account\_uuid`. Should be updated to utilize `crm\_account\_uuid` exclusively.

3. Downstream Analytics Models

- Model: `analytics.dim\_customer`
 - Risk: High. Uses `legacy\_account\_id` as a natural key in Slowly Changing Dimension (SCD) Type 2 merge. Dropping this will disrupt historical data management.
- Model: `analytics.fct\_subscriptions`
 - Risk: High. Joins on `legacy\_account\_id` to `dim\_customer`. This will affect subscription data integrity and reporting.
- Model: `analytics.customer\_360\_snapshot`
 - Risk: Medium. Uses `legacy\_account\_id` in the export to Salesforce. This feature will need rework to ensure continuity.
- Model: `analytics.churn\_training\_features`
 - Risk: Medium. Uses `legacy\_account\_id` as feature store entity id, requiring updates to maintain the integrity of churn analysis.

Migration Risks for Unmigrated Accounts

With 1,204 accounts still using `legacy\_account\_id`, dropping the column poses several risks:

- Data Loss: Unmigrated accounts will lose their identifiers if their transactions or profiles are still reliant on `legacy\_account\_id`.
- Operational Disruption: Systems relying on these accounts may face errors or face data inconsistency in reporting and analytics.
- User Experience Impact: If any customer-facing systems refer to these accounts, they may fail to retrieve essential customer data.

Recommended Migration Order

1. Immediate Action for Unmigrated Accounts:

- Prioritize migrating the remaining 1,204 legacy accounts to `crm\_account\_uuid` to minimize risk before the column drop.

2. Refactor Affected Models:

- Start with high-risk models:
 1. `analytics.dim\_customer`
 2. `analytics.fct\_subscriptions`
- Follow with medium-risk models:
 3. `analytics.customer\_360\_snapshot`
 4. `analytics.churn\_training\_features`
- Update the `staging.stg\_customers` and `intermediate.int\_customer\_identity` models last, as these are direct dependents and will be affected after the migration of downstream models.

Rollback Considerations

- Data Backup: Ensure that a backup of current data, particularly the `legacy\_account\_id` column, is available before making any changes.
- Version Control: Maintain version control of SQL scripts or transformation logic, so that if issues arise, the previous state can be reinstated.
- Testing Environment: Utilize a staging environment to simulate the drop before executing in production. Implement comprehensive testing across all affected models.
- Monitoring Post-Deployment: After deployment, closely monitor the models for any data integrity issues or breakdowns in

pipeline processing.

Timeline Management

Given the deadline of July 12, 2026, establish a phased approach that allows ample time for migration, testing, and rollback planning. A series of checkpoints should be scheduled to evaluate progress against the 94.2% CRM account migration completion and address any unforeseen challenges.

Engineering counterpart
2:48 AM - Responded in 0s

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

Agent response

2:48 AM - Responded in 8s

Certainly! Let's break down the reasoning step by step regarding the impact of dropping the `legacy\_account\_id` column and the considerations involved. Below are the considerations, assumptions, and reasoning behind each aspect:

Step-by-Step Reasoning

1. Understanding the Proposed Change:

- The action of dropping the `legacy\_account\_id` column is based on the premise that it has been deprecated due to a CRM migration, where it is replaced by `crm\_account\_uuid`.
- The significance of this change is to streamline data architecture by eliminating unnecessary legacy identifiers.

2. Identifying Direct Dependents:

- The models directly dependent on `legacy\_account\_id` were identified: `staging.stg\_customers` and `intermediate.int\_customer\_identity`.
- Assumption: These models are actively used for downstream processing and will be impacted by the removal of the column.
- It was presumed that any direct references to `legacy\_account\_id` in these models would break due to the column drop.

3. Assessing Downstream Marts/Models:

- Analyzed the downstream analytics models that reference `legacy\_account\_id`, assessing the risk associated with each based on how they use this column.
- Assigned risk levels (high/medium) based on:
 - Natural Keys: Models using the column as a natural key (e.g., `dim\_customer`) inherently have a high risk of breaking existing functionality.
 - Joins and Exports: The models that join or export data using `legacy\_account\_id` also carry significant risk, which can disrupt data integrity.

4. Evaluating Migration State:

- Noted that 94.2% of accounts had been migrated to `crm\_account\_uuid`, leaving 1,204 accounts still using `legacy\_account\_id`.
- Assumption: The remaining accounts are the ones that pose the highest risk if the column is dropped without migrating them first.

5. Migration Risks:

- Identified risks associated with unmigrated accounts. The primary concerns included data loss, operational disruptions, and potential user experience issues.
- Assumption: Any operational processes or reports relying on these accounts would fail or yield incorrect data post-column drop.

6. Recommending Migration Order:

- Suggested a phased approach to migration, prioritizing high-risk models first and then moving to the lower-risk models.
- The reasoning was that addressing the most critical dependencies first would reduce the overall risk of failure in the system.
- Each model's dependency hierarchy was considered to ensure that downstream dependencies are addressed in the correct order.

7. Rollback Considerations:

- Emphasized the importance of having a rollback plan in place, including data backups and version control.
- Assumption: Any change in a production environment must be reversible to ensure stability in operations if issues arise.
- Suggested thorough testing in a staging environment to ensure that any changes do not introduce unforeseen errors.

Key Assumptions Made

- Dependencies: The analysis assumed that the models listed as dependents are directly tied to the functioning of downstream applications and reports.

- Migration Completeness: It was assumed that if accounts are not yet migrated to `crm\_account\_uuid`, they will not be functional once `legacy\_account\_id` is removed.
- Risk Level Assignments: The risk levels assigned (high and medium) were subjective assessments based on the usage of the legacy column and potential impacts on data integrity.
- Existence of Backups: Assumed that there are established processes for data backup and restoration, which is critical for rollback scenarios.

Overall, the analysis hinges on understanding the data lineage and dependencies of `legacy\_account\_id`, assessing risks associated with unmigrated accounts, and ensuring that any changes introduced do not disrupt overall data integrity or operational functionality.

Engineering counterpart
2:48 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

2:48 AM - Responded in 11s

Before shipping the proposed change of dropping the `legacy\_account\_id` column to production, it is crucial to conduct a thorough verification process to minimize the risk of disruptions. Here's a structured list of what I would verify:

1. Data Migration Verification

- Account Migration Completion:

- Confirm that all accounts have been successfully migrated from `legacy\_account\_id` to `crm\_account\_uuid`, ensuring that the percentage has reached 100%.
- Validate that any unmigrated accounts (currently at 1,204) have been addressed or have a clear plan in place for migration before the column drop.

2. Dependency Check

- Model Impact Assessment:

- Reassess all models that reference `legacy\_account\_id` (both direct and indirect dependencies) to ensure they have been updated to use `crm\_account\_uuid` instead.
- Review any changes made to the `staging.stg\_customers` and `intermediate.int\_customer\_identity` models to confirm they no longer reference the legacy column.

3. Testing in Staging Environment

- Unit Tests:

- Execute unit tests for all modified models to ensure they function correctly with `crm\_account\_uuid` in place of `legacy\_account\_id`.
- Check that any business logic relying on these models produces expected results.

- Integration Tests:

- Perform integration testing to verify that the end-to-end flow of data remains intact and that downstream models (like `analytics.dim\_customer` and `analytics.fct\_subscriptions`) can process and utilize the data correctly without errors.

4. Data Validation

- Data Integrity Checks:

- Conduct data integrity checks to ensure that the migration has retained the correctness of data. Validate that data linked to `crm\_account\_uuid` matches that which had been linked to `legacy\_account\_id`.
- Compare sample outputs from dependent models before and after making changes to ensure consistency.

5. Monitoring and Rollback Plan

- Monitoring Setup:

- Ensure that monitoring systems are in place to track any errors or anomalies arising after the change is deployed. Set up alerts for any failed jobs or data discrepancies.

- Rollback Readiness:

- Confirm that a rollback plan is documented, including data backups of the current state before any change is made.
- Ensure that relevant teams are informed and prepared to execute the rollback if needed.

6. Stakeholder Communication

- Communication Plan:

- Inform all stakeholders (data consumers, business units, etc.) about the planned change. Define timelines and expected impacts on their operations.
- Prepare documentation on the migration process, changes made, and any necessary adjustments that users may need to implement in their queries or reports.

7. Performance Testing

- Performance Impact Analysis:

- Assess performance impacts of the new structure. Validate that the changes do not negatively affect query performance or system throughput.

- Running performance benchmarks and load tests can provide insights into any potential issues that may arise post-deployment.

8. Final Review

- Code Review:

- Conduct a peer review of all code changes made during the migration and adjustment process.
- Ensure that all changes comply with coding standards and best practices.

By verifying these elements, you can ensure that the transition from ``legacy_account_id`` to ``crm_account_uuid`` is seamless, minimizes risk, and maintains data integrity across the system once the change is deployed to production.

Engineering counterpart
2:48 AM - Responded in 0s

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

Agent response

2:48 AM - Responded in 19s

When considering the change of dropping the `legacy\_account\_id` column from the `raw.customers` table, several top risks must be taken into account, along with a comprehensive rollback plan in case of failure. Here's an overview:

Top Risks

1. Data Loss for Unmigrated Accounts:

- Risk: If `legacy\_account\_id` is dropped before all accounts are migrated to `crm\_account\_uuid`, any transactions or references to these unmigrated accounts will result in data loss.
- Mitigation: Ensure that all accounts are migrated and confirm that a plan for unmigrated accounts is in place prior to making changes.

2. Model Breakage:

- Risk: Downstream models that reference `legacy\_account\_id` may break if they are not properly updated to use `crm\_account\_uuid`, resulting in errors or failure in processing.
- Mitigation: Thoroughly test all dependent models in a staging environment and ensure they are verified before deployment.

3. Loss of Historical Data Integrity:

- Risk: The removal of the legacy column could affect historical records if it is used as a natural key in Slowly Changing Dimensions (SCDs).
- Mitigation: Modify SCD implementations to use `crm\_account\_uuid` and validate that historical data can still be accessed and reported accurately.

4. User Experience Impact:

- Risk: If user-facing systems rely on `legacy\_account\_id`, dropping the column could lead to errors or incomplete information being displayed.
- Mitigation: Communicate changes to stakeholders and ensure systems are updated accordingly before deployment.

5. Performance Issues Post-Deployment:

- Risk: Changes might introduce performance degradation in queries or data processing due to unexpected data structures.
- Mitigation: Conduct performance testing pre- and post-deployment to identify and address issues.

Rollback Plan

In the event that the deployment does not go as planned, a comprehensive rollback plan should be established as follows:

1. Data Backup:

- Action: Prior to making any changes, take a complete backup of the `raw.customers` table, including the `legacy\_account\_id` column.
- Location: Store backups in a secure location that allows for easy access in case of rollback.

2. Version Control:

- Action: Maintain a copy of the SQL scripts or migration logic used to drop the column, along with the previous versions of models.
- Restore Points: Versioning provides a way to revert to the previous codebase.

3. Rollback Execution:

- If issues arise after deployment:
 - Restore Data: Use the backup to restore the `legacy\_account\_id` column in `raw.customers`.
 - Revert Models: Redeploy the previous versions of all dependent models that referenced `legacy\_account\_id`.
 - Adjust Business Logic: Ensure all queries and business processes that depend on `legacy\_account\_id` are restored to their prior state.

4. Validation Post-Rollback:

- Data Integrity Checks: After restoring, validate the integrity of the data to ensure that it matches the state before the change.
- Functionality Testing: Test all impacted models, reports, and user-facing systems to confirm that functionality is restored and there are no residual issues.

5. Communication:

- Inform Stakeholders: Notify all relevant teams and stakeholders of the rollback action and the reasons behind it.
- Document Findings: Document any issues encountered during the deployment to inform future change management processes.

By understanding these risks and having a clear rollback plan in place, you can approach the change with a structured strategy that prioritizes data integrity and operational continuity.

Engineering counterpart
2:48 AM - Responded in 0s

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

Agent response
2:49 AM - Responded in 11s

Certainly! Below is a concise action plan outlining concrete next steps, assigned owners, and proposed timings for each task related to dropping the `legacy\_account\_id` column from the `raw.customers` table.

Next Steps Action Plan

| Step | Description | Owner | Timing |
|------|---|----------------------------|---|
| 1 | Complete Account Migration | Data Migration Team | By 2024-01-15 |
| | Confirm that 100% of accounts have been successfully migrated to `crm_account_uuid`. Verification should include addressing any remaining 1,204 legacy-only accounts. | | |
| 2 | Update Dependent Models | Data Engineering Team | By 2024-02-15 |
| | Refactor all models that reference `legacy_account_id` to use `crm_account_uuid`. This includes `staging.stg_customers`, `intermediate.int_customer_identity`, and all downstream analytics models. | | |
| 3 | Testing in Staging Environment | Data QA Team | By 2024-03-01 |
| | Conduct unit tests and integration tests on all modified models in a staging environment. Validate that all functionalities work as expected without referencing `legacy_account_id`. | | |
| 4 | Conduct Performance Testing | Data Engineering & QA Team | By 2024-03-10 |
| | Perform performance benchmarking and load tests to ensure that there are no performance degradations in the system after changes are made. | | |
| 5 | Backup Existing Data | Data Engineering Team | By 2024-03-15 |
| | Take a complete backup of the `raw.customers` table, including the `legacy_account_id` column. Ensure data is stored securely for potential rollback. | | |
| 6 | Communication with Stakeholders | Project Manager | By 2024-03-20 |
| | Inform all relevant stakeholders about the upcoming changes, including the timeline, potential impacts, and necessary adjustments they may need to make. Prepare documentation outlining the changes. | | |
| 7 | Deploy Changes to Production | Data Engineering Team | By 2024-03-25 |
| | Execute the removal of the `legacy_account_id` column from the `raw.customers` table and update all affected models in the production environment. | | |
| 8 | Monitoring Post-Deployment | Data Engineering & QA Team | From 2024-03-26 onward |
| | Monitor system logs and data processing workflows for errors or issues. Ensure that alerts are set up for any discrepancies in data integrity. | | |
| 9 | Rollback Plan Execution (if needed) | Data Engineering Team | As needed, immediately following deployment |
| | If critical issues are identified post-deployment, restore the `legacy_account_id` column from the backup and revert all dependent models to their prior state. Document findings and issues encountered during the deployment. | | |

Summary

This action plan outlines clear ownership and deadlines for each critical step in the process leading up to and following the removal of the `legacy\_account\_id` column. By adhering to this structured approach, the team can effectively manage the transition while minimizing risks and ensuring data integrity throughout the process.

Engineering counterpart
2:49 AM - Responded in 0s

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

Agent response

2:49 AM - Responded in 5s

The one thing I would want a second pair of eyes on is the comprehensive testing strategy for all modified models and dependent systems in the staging environment before deployment to production.

Reason for Focus:

- Criticality of Testing: Given the significant risk associated with dropping the `legacy\_account\_id` column, it is essential to ensure that every model that previously relied on this column has been thoroughly tested to confirm functionality and data integrity using `crm\_account\_uuid`.
- Integration and End-to-End Testing: Ensuring that all pieces work together correctly is vital, especially since several downstream processes may be affected. I would want to confirm that all scenarios have been tested, including edge cases, to avoid unexpected failures post-deployment.
- Validation of Migration Logic: It's also crucial to validate that the migration logic for accounts from `legacy\_account\_id` to `crm\_account\_uuid` is working as intended and that there are no discrepancies in the transformed data.

Key Aspects to Review:

1. Test Coverage: Verify that all relevant models, queries, and reporting functionalities have been included in the testing plan.
2. Test Case Documentation: Ensure that clear documentation of test cases, expected outcomes, and actual results is maintained for accountability.
3. Performance Testing: Review the plan for performance benchmarks to check for any degradation in system performance post-deployment.
4. Rollback Testing: Ensure that rollback procedures have been tested in the staging environment to minimize risks associated with production failures.

Having an additional perspective on this aspect can provide further assurance that we are adequately prepared for deployment and can help identify any gaps or oversights before proceeding.

Engineering counterpart

2:49 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.