



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: qwen/qwen3-32b

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

Duration: 3m 33s

Actual run cost: \$0.001308

Projected execution cost: \$0.2878/mo (10/biz day × 22 days)

Report date: July 14, 2026 - 3:08 AM EST

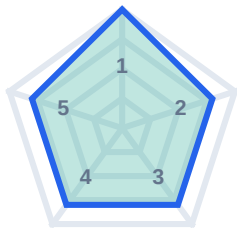
Overall

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: 13543 chars - Latency: 213s - 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: 3m 33s

Engineering counterpart

7:04 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

7:05 AM - Responded in 48s

Lineage Impact Review for `raw.customers.legacy\_account\_id` Retirement

1. Affected Models/Marts

High-Risk

- `analytics.dim\_customer`
 - Critical usage in SCD2 merge logic (natural key). Disruption could corrupt historical customer records.
- `analytics.fct\_subscriptions`
 - Requires join on `legacy\_account\_id` to `dim\_customer`. Failure would break subscription linkages and reporting.

Medium-Risk

- `analytics.customer\_360\_snapshot`
 - Legacy ID used in Salesforce exports. Requires validation that UUID-based identifiers can replace legacy IDs for integration.
- `analytics.churn\_training\_features`
 - Entity ID in ML feature store. Model performance may degrade if IDs are missing or inconsistent.

Dependent Models

- `staging.stg\_customers`: Stores raw customer data with legacy column. Requires replacement with `crm\_account\_uuid`.
- `intermediate.int\_customer\_identity`

Engineering counterpart

7:05 AM - Responded in 0s

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

Agent response

7:06 AM - Responded in 58s

Detailed Reasoning & Assumptions for `legacy\_account\_id` Retirement Review

1. Affected Models/Marts

High-Risk Models

- `analytics.dim\_customer` (SCD2 merge)

- Assumption: Legacy ID was the natural key for identifying customer records in a Slowly Changing Dimension Type 2 (SCD2) table.

- Implications:

- SCD2 tables store historical changes using natural keys (e.g., legacy IDs). Dropping `legacy\_account\_id` would break the ability to perform SCD2 merges, leading to potential data corruption in historical records.

- If the key is replaced with `crm\_account\_uuid`, existing historical records must be updated to reflect the new UUIDs to maintain consistency.

- Validation Needed: Confirm if `crm\_account\_uuid` is stable and can replace legacy IDs in SCD2 logic without causing unintended duplicates or record merges.

- `analytics.fct\_subscriptions` (join on legacy ID)

- Assumption: The fact table's join logic to `dim\_customer` relies on `legacy\_account\_id` for stitching transactional data (e.g., subscriptions) to customer attributes.

- Implications:

- A missing join key would result in incomplete or inaccurate subscription data, breaking downstream reporting (e.g., customer lifetime value, churn rates).

- If `crm\_account\_uuid` is used inconsistently in the fact table, joins to `dim\_customer` will fail for records where the UUID is missing.

- Validation Needed: Ensure all fact table records can be mapped to UUIDs, especially the 1,204 legacy-only accounts.

Medium-Risk Models

- `analytics.customer\_360\_snapshot` (Salesforce export)

- Assumption: Salesforce requires `legacy\_account\_id` to map customer data for integrations (e.g., syncing with CRM systems).

- Implications:

- If Salesforce hasn't been updated to use UUIDs, this export will fail or include invalid data.

- Salesforce may rely on legacy IDs as external identifiers that cannot be changed.

- Validation Needed: Confirm whether Salesforce can be updated to accept UUIDs. If not, a temporary mapping or dual-identifier strategy may be required (e.g., join UUIDs to legacy IDs using `int\_customer\_identity`).

- `analytics.churn\_training\_features` (feature store entity ID)

- Assumption: The churn model uses `legacy\_account\_id` as a primary identifier to link features (e.g., historical behavior) to customers.

- Implications:

- Missing entity IDs could result in incomplete training data for 1,204 legacy-only accounts.

- UUIDs must be stable and deterministic to avoid "identity shift" in ML models (e.g., 1204 accounts might appear as new entities, skewing training).

- Validation Needed: Ensure UUIDs are consistent for legacy-only accounts (e.g., via data validation rules). If not, consider how to handle "identity shift" in the ML pipeline.

Dependent Models

- `staging.stg\_customers`

- Assumption: This model is the first layer of transformation for `raw.customers`. Dropping the legacy column would break its transformation logic unless it's updated.

- Action Required: Remove all references to `legacy\_account\_id` in the model's SQL/DDDL and ensure fallback logic does not rely on the column.

- ``intermediate.int_customer_identity``

- Assumption: This model likely handles identity mapping (e.g., linking legacy IDs to UUIDs). The fact that it references both columns suggests it's part of the

Engineering counterpart
7:06 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:06 AM - Responded in 15s

Pre-Production Verification Checklist for `raw.customers.legacy\_account\_id` Retirement

Before dropping the column, I'd validate the following to minimize risk and ensure alignment with the CRM cutover deadline (2026-07-12):

1. Data Completeness & Coverage

- UUID Population Rate:
 - Confirm that the `crm\_account\_uuid` is fully populated (100%) in the `raw.customers` table for all 1204 legacy-only accounts by the deadline.
 - If 5.8% (1204 accounts) are still missing UUIDs, verify whether:
 - These accounts will be migrated during CRM's cutover.
 - A temporary fallback strategy exists (e.g., mapping UUIDs to legacy IDs via `int\_customer\_identity` for backward compatibility).
- Consistency Between Legacy ID and UUID:
 - Validate that `crm\_account\_uuid` is deterministic and stable for all customers (including legacy-only accounts). For SCD2 and ML models, UUIDs must not change during migration.
 - Cross-check UUIDs in `raw.customers` against `int\_customer\_identity` to ensure there are no discrepancies or missing mappings.

2. Model/Query Logic Validity

- SCD2 Merge in `dim\_customer`:
 - Ensure the natural key logic for SCD2 is updated to use `crm\_account\_uuid` instead of `legacy\_account\_id`.
 - Test historical SCD2 records to confirm no duplicates or inconsistencies arise from the switch (e.g., UUIDs for legacy accounts must align with existing surrogate keys).
- Fact Table Joins in `fct\_subscriptions`:
 - Validate that all `fct\_subscriptions` records can now join to `dim\_customer` using `crm\_account\_uuid`.
- Test edge cases:
 - Subscriptions with `crm\_account\_uuid` = NULL (if any).
 - Legacy-only accounts being mapped via `int\_customer\_identity` (if temporary).
- Salesforce Export in `customer\_360\_snapshot`:
 - Confirm Salesforce systems can accept `crm\_account\_uuid` as the identifier (e.g., field types match, external IDs are updated).
 - If not, ensure `int\_customer\_identity` is active and used as a bridge to map UUIDs -> legacy IDs for the export.
- ML Feature Store in `churn\_training\_features`:
 - Verify that all legacy-only accounts have their UUIDs backfilled or mapped in the training data.
 - Test model behavior with UUIDs to ensure no identity shift (e.g., 1204 accounts should not appear as "new" entities post-migration).

3. Staging/Intermediate Layer Readiness

- `staging.stg\_customers`:
 - Ensure the model no longer references `legacy\_account\_id`.
 - Validate that all downstream transformations (e.g., `int\_customer\_identity`) can proceed without the column.
- `intermediate.int\_customer\_identity`:
 - Confirm the model is actively resolving legacy-to-UUID mappings for unmigrated accounts.
 - Test if the UUID is the primary identifier in downstream models (e.g., `dim\_customer`, `fct\_subscriptions`).

4. Data Pipeline Testing

- Backfill Legacy Data:
 - Ensure historical data in `dim\_customer` and `fct\_subscriptions` has been retrofitted with `crm\_account\_uuid` via ETL processes.
- Schema Changes in Staging Models:
 - Run a test pipeline with the column removed from `raw.customers`.
 - Validate that `staging.st

Engineering counterpart
7:06 AM - Responded in 0s

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

Agent response

7:07 AM - Responded in 34s

Top Risks of Proceeding

1. Historical Data Corruption in `dim\_customer` (SCD2)

- Risk: If `legacy\_account\_id` is dropped before all historical records in `dim\_customer` are migrated to use `crm\_account\_uuid`, SCD2 logic may fail to track historical changes correctly, leading to inconsistent or duplicate records.
- Criticality: High (affects all customer history).

2. Break in `fct\_subscriptions` Join Logic

- Risk: If `fct\_subscriptions` cannot join to `dim\_customer` (due to missing UUIDs for legacy-only accounts), subscription data will be missing from reporting and analytics.
- Criticality: High (impacts revenue, churn, and customer value metrics).

3. Unmigrated Accounts (1204 legacy-only accounts)

- Risk: These accounts will lose their identifiers entirely if `legacy\_account\_id` is dropped and `crm\_account\_uuid` is not backfilled.
- Criticality: High (data loss for 5.8% of customer records).

4. Salesforce Integration Failure

- Risk: If Salesforce has not been updated to accept `crm\_account\_uuid`, the export in `customer\_360\_snapshot` will fail, disrupting downstream business processes (e.g., sales operations).
- Criticality: Medium (impacts non-critical workflows if temporary workarounds exist).

5. ML Model Degradation

- Risk: Churn models using `legacy\_account\_id` as an entity key may lose historical alignment for unmigrated accounts, causing incorrect predictions or retraining delays.
- Criticality: Medium (impacts ML accuracy if UUIDs are not retroactively mapped).

Rollback Plan

If the column drop causes critical failures, the rollback must restore the original state and ensure downstream dependencies are fixed. Here's the structured approach:

1. Immediate Reversal (Pre-Column Drop)

- If the column is not yet dropped:
 - Cancel the schema change and revert the DDL/DDL scripts.
 - Validate that all dependencies continue to use `legacy\_account\_id` as-is.

2. If the Column Already Dropped

- Step 1: Reintroduce the Column Temporarily
 - Add `legacy\_account\_id` back to `raw.customers` (assuming it can be restored from backups or source systems).
 - If source data is lost:
 - Use the `int\_customer\_identity` model (which maps legacy IDs to UUIDs) to populate `legacy\_account\_id` in downstream models temporarily.
 - Manually backfill missing data for 1204 accounts if necessary (e.g., from CRM migration logs).
- Step 2: Revert Dependent Models
 - `staging.stg\_customers`:
 - Restore `legacy\_account\_id` to the model's SQL.
 - `analytics.dim\_customer`:
 - Reintroduce `legacy\_account\_id` as the natural key in SCD2 merge logic.
 - `analytics.fct\_subscriptions`:
 - Revert joins to use `legacy\_account\_id`.
 - `int\_customer\_identity`:

- Ensure legacy ID <-> UUID mappings remain active for future use.

- Step 3: Monitor and Test

- Re-enable automated alerts for missing UUIDs in other areas (e.g., `fct\_subscriptions` should flag any new unmigrated accounts post-rollback).

- Run regression tests to confirm historical data integrity and joins are restored.

3. Post-Rollback Mit

Engineering counterpart

7:07 AM - Responded in 0s

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

Agent response

7:07 AM - Responded in 13s

Concrete Next Steps with Owners, Deadlines, and Key Actions

1. Final Data Validation & Backfilling

Owner: Data Engineering Team

Deadline: June 2026 (1 month before CRM cutover)

- Action:

- Validate all 1,204 legacy-only accounts have `crm\_account\_uuid` backfilled or mapped via `int\_customer\_identity`

Engineering counterpart

7:07 AM - Responded in 0s

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

Agent response

7:08 AM - Responded in 45s

The one critical item I'd want a second pair of eyes on is:

Key Peer-Review Focus

Component: SCD2 Merge Logic in `analytics.dim\_customer` when switching from `legacy\_account\_id` to `crm\_account\_uuid`

Owner: Data Engineering Team (SCD2 specialist)

Deadline: At least 1 week before column drop

Why it matters:

- SCD2 is complex because it tracks historical changes in customer attributes. A wrong implementation (e.g., duplicate UUIDs, mismatched identifiers) could silently corrupt historical records, making it impossible to trace changes post-migration.
- Legacy systems often rely on SCD2 for reporting, audits, and compliance. Even small errors here have long-term, unrepairable impacts.

What to verify:

1. UUID Stability: Confirm `crm\_account\_uuid` never changes for a given customer (even during migration).
2. Surrogate Key Consistency: Ensure SCD2 logic does not create new surrogate keys for existing customers when switching to UUIDs.
3. Historical Mapping: Validate that all historical `legacy\_account\_id` records have been mapped to UUIDs in `int\_customer\_identity` and that joins in `fct\_subscriptions` work post-migration.
4. Edge Case Handling: Test for legacy-only accounts (`1204` in `raw.customers`) to ensure UUIDs are correctly assigned now (not left undefined).

Action:

- Have a SCD2 expert or lead data modeler review the ETL logic for `dim\_customer` and `int\_customer\_identity`, ideally with a code freeze/pull request review process.
- Run regression tests on historical data (e.g., "Did Customer X have this attribute on Date Y?") using both legacy and UUID keys to confirm accuracy.

This step is critical because SCD2 data corruption is irreversible and not easily detectable without rigorous validation. If this is flawed, the rollback would require manual data rehydration and could still leave gaps in historical timelines.

Engineering counterpart

7:08 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.