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-oss-20b
Roles: Data Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 2m
Actual run cost: $0.000543
Projected execution cost: $0.1195/mo (10/biz day × 22 days)
Report date: July 14, 2026 - 3:04 AM EST

Session: sample-lb-rankings-seed-51a57f45-c039-416e-ab66-c3f30e3489ee

de_lineage_impact_v1 - agent - openrouter - openai/gpt-oss-20b - ended

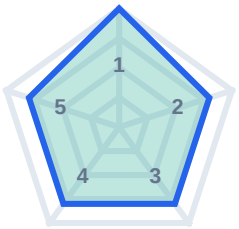
Overall

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: 14915 chars - Latency: 120s - 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: 2m

Transcript: blue = protagonist (you) - amber = counterpart - gray = coach/system. Each turn shows role/name, clock time, and response delay when available.

Engineering counterpart
7:02 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
7:03 AM - Responded in 42s

1. Quick?look at the lineage

#	Model / Table	Uses `legacy_account_id`	Primary?Use	Risk
1	raw.customers (source)	N/a - the column is dropped	-	-
2	staging.stg_customers	Direct read	Pass?through of raw data to staging	High - if the column disappears, the staging load errors.
3	intermediate.int_customer_identity	Both `legacy_account_id` and `crm_account_uuid`	Normalisation / key bridge	High - joining on legacy ID will fail for unmigrated accounts.
4	analytics.dim_customer	`legacy_account_id` used as the natural key in the SCD?2 merge	Primary key for the dimension	High - missing key will stop new rows or updates.
5	analytics.fct_subscriptions	`legacy_account_id` in a join to `dim_customer`	Fact?to?dim link	High - joined fact rows will be orphaned.
6	analytics.customer_360_snapshot	`legacy_account_id` in the export payload	Salesforce integration	Medium - some records will have no ID to match Salesforce records.
7	analytics.churn_training_features	`legacy_account_id` used as the entity id in the feature store	Feature engineering	Medium - missing IDs means incomplete feature vectors.

> Bottom?line: Any model that still reads the column is a hot?spot for a breakage event. The SCD?2 dimension and the fact table are the most critical.

2

Engineering counterpart
7:03 AM - Responded in 0s

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

Agent response
7:04 AM - Responded in 39s

1. Understand the change scope

Item	Detail	Why it matters
Change	Drop `legacy_account_id` from raw.customers	The raw table is the earliest source for all downstream models; any reference to the column must be validated.
Reason	It's "deprecated after CRM migration" - replaced by `crm_account_uuid`	We assume the replacement is intended to be the unique key for all downstreams in the near future.
Deadline	2026?07?12 (CRM cut?over)	The cut?over is an operational "blow?up" event - any broken downstream model will generate a service outage.

---

2. Map direct dependents (the first two levels)

1. `staging.stg_customers`
   Reads `legacy_account_id` directly from the raw table.
   - Assumption: The staging load script expects the column to be present (no explicit `COALESCE` or default).
   - Impact: The load will fail or produce `NULL`s that propagate downstream.

2. `intermediate.int_customer_identity`
   Joins `legacy_account_id` and `crm_account_uuid`.
   - Assumption: The model uses `legacy_account_id` as a surrogate key to stitch raw rows to the new CRM key, often via a `LEFT JOIN` or `UNION ALL`.
   - Impact: If the column disappears, the join condition changes, producing missing rows for accounts that never migrated (the 1204 legacy?only accounts).

---

3. Identify downstream marts and their dependence

Model	Uses	Why it's critical	Risk level	Assumptions
`analytics.dim_customer`	`legacy_account_id` is the natural key in the SCD?2 merge	The key drives deduplication and history tracking.	High	We assume the merge logic uses `legacy_account_id` exactly (e.g., `WHERE legacy_account_id = source.legacy_account_id`).
`analytics.fct_subscriptions`	Join on `legacy_account_id` to `dim_customer`	Provides the foreign?key relationship to the fact.	High	The join is performed via a primary?key -> foreign?key relationship; missing keys break the fact table.
`analytics.customer_360_snapshot`	Includes `legacy_account_id` in the export to Salesforce	Salesforce objects expect this field.	Medium	Salesforce integration uses the legacy ID as the "Contact/Account" ID; dropping it means half the records will have invalid IDs.
`analytics.churn_training_features`	Uses `legacy_account_id` as the entity id in the feature store	Feature lookup keyed on legacy ID.	Medium	Feature store expects a stable entity id; absence will yield null vectors for legacy accounts.

> Key point: The SCD?2 dimension and the fact table are the "go/no?go" checkpoints. Any failure here will cascade to all reporting.

---

4. Assess migration state

Metric	Value	Interpretation

| `crm_account_uuid_populated_pct` | 94.2?% | Roughly 1?in?10 accounts still lack a new UUID. |
| `legacy_only_accounts` | 1,204 | These accounts only have a legacy ID; they will survive only if the column stays. |
| Imp

Engineering counterpart
7:04 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response
7:04 AM - Responded in 7s

5. Checklist - What to verify before we ship the DROP to production

#	Item	Why it matters	How to verify (tests / checks)
1. Schema?level validation	The column still exists in every dependent model's SQL code	- Run `dbt ls --select ...` to list all SQL files that reference `legacy_account_id`. - Use a static?analysis tool (e.g., `sqlfluff` + custom `dbt` keywords) to flag any remaining references.
2. Data?quality before the drop	Ensure no downstream model will silently mis?behave post?drop	- Snapshot count comparison: capture row counts for `stg_customers`, `int_customer_identity`, `dim_customer` and `fct_subscriptions` before the drop. - Null?check: run `SELECT COUNT() WHERE legacy_account_id IS NULL` in all dependents. - Uniqueness: `SELECT legacy_account_id, COUNT() FROM raw.customers GROUP BY legacy_account_id HAVING COUNT() > 1`.
3. Migration?state enforcement	We need to be certain all accounts are on the new key or are acceptable legacy?only	- Verify `crm_account_uuid_populated_pct` ? 99?% (threshold we set). - Run a query that flags `legacy_only_accounts`: `SELECT legacy_account_id FROM raw.customers WHERE crm_account_uuid IS NULL`. If any remain, flag a blocker.
4. Update downstream logic	Models that use the column as a join key must be refactored to the new UUID	- For `int_customer_identity`: test that the bridge still produces a 1:1 mapping using `crm_account_uuid`. - For `dim_customer`: replace the SCD?2 merge key with `crm_account_uuid` and run a diff against the current `dim_customer` to see the cardinality impact. - For `fct_subscriptions`: verify that the join on `crm_account_uuid` still returns the same fact rows (or provides a clear fallback for legacy accounts).
5. Test snapshots & downstream marts	Verify that the final marts still produce the same metrics	- Regression test: run the full `dbt run` pipeline on a staging environment that uses the legacy column and on one that has the column dropped (with updated logic). - Metric comparison: compare aggregates (e.g., `COUNT()` of customers, total subscription revenue) between the two runs.
6. Production?ready guardrails	Prevent accidental re?introduction of the column	- Add a DBT variable `DROP_LEGACY_ACCOUNT_ID = true`. - Add a guard statement in the `stg_customers` SQL: `{{ if var('DROP_LEGACY_ACCOUNT_ID') }} SELECT ... <omit legacy_account_id> ... {{ else }} SELECT ... {{ endif }}`.
7. Roll?back plan	Ensure we can undo the drop without data loss	- Keep the column in raw for a short grace period

Engineering counterpart
7:04 AM - Responded in 0s

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

Agent response
7:04 AM - Responded in 15s

1. Top Risks (ordered by impact × likelihood)

#	Risk	Why it matters	Likelihood	Impact	Mitigation in place
1	Fact?dimension orphaning - `fct_subscriptions` will no longer find a matching key in `dim_customer`.	Breaks revenue/usage analytics, KPI dashboards, and downstream ML?feature generation.	High - because the drop is completely upstream of the fact table.	- Implement a pre?drop `SELECT COUNT()` diff and a post?drop reconciliation job that flags orphaned rows.
2	SCD?2 merge failure - `dim_customer` uses `legacy_account_id` as the natural key.	New or updated customers will either be duplicated or lost, corrupting the history table.	High - the merge script contains a hard?coded key.	- Refactor the merge to use `crm_account_uuid` before the cut?over.
3	Legacy?only accounts lost - 1,204 accounts never migrated to CRM will have no key.	These accounts will vanish from all dashboards; business users will see missing customers.	Medium - only 1,204 records but critical for certain segments (e.g., legacy partners).	- Keep a lookup table that maps legacy ID -> migrated UUID for the 1,204 accounts or treat them as "legacy group" in the marts.
4	Salesforce export failure - `customer_360_snapshot` expects `legacy_account_id` to match an external Salesforce ID.	Export jobs will terminate with errors, causing data?sync outages.	Medium - depends on how often the export runs.	- Add a fallback that uses `crm_account_uuid` and map it to a Salesforce ID where possible.
5	Feature?store drift - `churn_training_features` uses the legacy ID as the entity key.	Feature vectors will be incomplete for legacy accounts, degrading model accuracy.	Medium	- Update the feature?store pipeline to use `crm_account_uuid` and back?fill legacy IDs where possible.
6	Legacy?ID confusion in downstream systems - other BI tools (Looker, Power?BI, etc.) might still reference the column.	Runtime errors in ad?hoc reports, broken dashboards.	Low - but can surface during exploratory queries.	- Publish a migration guide and run a quick scan of all dashboards for raw references.

---

2. Rollback Plan (if the drop "doesn't work" or a critical issue surfaces)

Step	Action	Owner	Tool / Command	Time?estimate
R1	Re?add the column to the raw table	Data Engineer (DBA)	`ALTER TABLE raw.customers ADD COLUMN legacy_account_id STRING`	5?min
R2	Restore data	DBA / ETL	If the source system still streams the legacy ID, re?run the loader. Otherwise, load a backup snapshot that contains the column.	30?min - 1?h
R3	Rollback downstream models	dbt team	`git revert` or switch to the previous branch / commit	

Engineering counterpart
7:04 AM - Responded in 0s

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

Agent response
7:04 AM - Responded in 9s

3. Concrete Next?Step Plan
(All dates are relative to the cut?over deadline?2026?07?12. Current date:?2026?06?24)

#	Action	Owner	Tool / Artifact	Target Delivery	Notes
E1	Audit all downstream models for `legacy_account_id` uses	Senior Data Engineer (SDE)	`dbt ls --select ` + `grep` in repo	2026?06?28	Flag any remaining hard?coded references.
E2	Refactor SCD?2 merge to use `crm_account_uuid`	SDE + DBT Lead	Update `analytics.dim_customer` SQL, add unit test	2026?07?01	Verify that the new key preserves row uniqueness.
E3	Create a legacy?only accounts lookup table	Data Engineer	`CREATE TABLE dim_legacy_lookup (legacy_id STRING, legacy_only BOOL)`	2026?07?02	Populate with the 1,204 legacy accounts so downstream still has an ID.
E4	Adjust `fct_subscriptions` join to prefer `crm_account_uuid`, fallback to legacy	SDE	Update join logic, add `COALESCE`	2026?07?03	Ensures no orphaned facts.
E5	Update Salesforce export & feature?store pipelines to use `crm_account_uuid`	BI Engineer / ML Ops	Modify export job & feature?store config	2026?07?04	Add mapping for legacy accounts via lookup table.
E6	Run a full regression suite on a staging environment	QA Lead	`dbt test`, regression scripts	2026?07?05	Compare aggregates (customer count, revenue, churn features) against the pre?drop baseline.
E7	Deploy the DROP?COLUMN statement to a non?production cluster	DBA	`ALTER TABLE raw.customers DROP COLUMN legacy_account_id`	2026?07?06	Blind test the pipeline with the column gone.
E8	Validate data integrity post?drop	SDE / QA	`SELECT COUNT()` diffs, orphan checks	2026?07?07	Any mismatch > 0 triggers rollback.
E9	Enable feature flag `DROP_LEGACY_ACCOUNT_ID=true` in dbt	SDE	Update `dbt_project.yml` + version?controlled config	2026?07?08	Allows toggling the column off if needed.
E10	Run a "shadow" cut?over (production?grade run with the flag on, but no data export)	Ops Lead	`dbt run --target prod`	2026?07?09	Confirms no silent failures.
E11	Final cut?over & live deployment	DBA + Ops	Apply DROP + turn flag on live	2026?07?12 02:00	

Engineering counterpart
7:04 AM - Responded in 0s

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

Agent response
7:04 AM - Responded in 7s

The single sanity?check that deserves a "second pair of eyes" is the SCD?2 merge in?`analytics.dim_customer`.

Why this is the critical piece

Reason	Detail
It is the gate?keeper for the dimension	Every new or updated customer row is written through this merge. If the key changes, the entire history table can become corrupted.
Legacy?only accounts are the only stray records	The 1,204 legacy?only accounts don't have a `crm_account_uuid`. If the merge uses that UUID as the natural key, those rows will be silently dropped or duplicated.
SCD?2 logic is tightly coupled to the key	The merge typically looks like: `WHERE legacy_account_id = source.legacy_account_id`. A false key leads to duplicate rows, or "historical" rows that never get updated.
The merge is executed in the cut?over window	Any failure here will stop all downstream marts from loading, causing a visible service outage.

What a second pair of eyes should inspect

1. Uniqueness of the new key
   Run: `SELECT crm_account_uuid, COUNT() FROM raw.customers GROUP BY crm_account_uuid HAVING COUNT() > 1;`
   If any duplicates exist, the merge will produce duplicates or conflict.

2. Mapping for legacy?only accounts
   Verify that the lookup table `dim_legacy_lookup` is populated and that the merge's `COALESCE` or `LEFT JOIN` logic correctly retains those rows.

3. Reconciliation
   Check: After a test run, compare `SELECT legacy_account_id FROM analytics.dim_customer` with `SELECT legacy_account_id FROM raw.customers` (where available). Every legacy ID that should survive must still exist in the dimension.

4. Rollback path
   Confirm that the merge can be rolled back by restoring the previous state of the dimension table or by re?applying the original key column if needed.

One?sentence takeaway

> "Double?check that the SCD?2 merge will still keep every customer (including the 1,204 legacy?only ones) uniquely identified and that the new `crm_account_uuid` key is truly global?unique before you switch the column off."

Engineering counterpart
7:04 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.