



Vantage RuntimeAI

Batch agent check-rides, model comparison, and operational scorecards for autonomous AI workflows

8.0/10

Overall score

Run summary

Scenario: Freshness SLA Breach

Model: openai/gpt-oss-120b

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

Duration: 1m 2s

Actual run cost: \$0.000460

Projected execution cost: \$0.5060/mo (50/biz day × 22 days)

Report date: July 14, 2026 - 2:56 AM EST

Overall

8.0/10

PASS - trust=high

Pass - score 8.0/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



#	Dimension (matches spoke #)	Score
1	Stale upstream identified	8/10
2	Uses SLA / freshness facts	8/10
3	Downstream blast radius	8/10
4	Structured status summary	8/10
5	Restore & verify steps	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.

- Upstream blocker identified: Met
- SLA breach acknowledged: Met
- Downstream impact noted: Met
- Restore steps listed: 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: 4 - Counterpart msgs: 5 - Output: 11282 chars - Latency: 62s - Criteria: 4/4

Score analysis

Identifies int_order_enriched failure as upstream blocker.

Score snapshot

Solid: Solid technical task response with a few gaps.

Interpretation

Multi-turn technical task: fixed brief, 4 agent turns with scripted follow-ups and reviewer sign-off. 4 of 4 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

- + Stale upstream identified: Correctly identifies the primary technical issue from the artifacts.
- + Uses SLA / freshness facts: Uses tables, logs, or schema facts from the task brief.
- + Downstream blast radius: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Structured status summary: Clear structure with readable SQL, diff, or memo formatting.
- + Restore & verify steps: Strong rationale and actionable follow-through.

DIMENSION NOTES

- Stale upstream identified: Correctly identifies the primary technical issue from the artifacts.
- Uses SLA / freshness facts: Uses tables, logs, or schema facts from the task brief.
- Downstream blast radius: Shows appropriate safety, budget, or risk awareness for the scenario.
- Structured status summary: Clear structure with readable SQL, diff, or memo formatting.
- Restore & verify steps: 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

Upstream blocker identified: yes - SLA breach acknowledged: yes - Downstream impact noted: yes - Restore steps listed: yes

CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

Transcript

Session

Scenario: Freshness SLA Breach

Agent response <-> Engineering counterpart

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

Duration: 1m 2s

Engineering counterpart

6:55 AM

Task brief - Freshness SLA breach (moderate complexity)

A P1 mart breached its freshness SLA. Triage using the metadata JSON:

```
```json
{
 "mart": "analytics.fct_orders",
 "owner_team": "data-platform",
 "sla": {
 "max_staleness_minutes": 120,
 "tier": "P1 revenue dashboard"
 },
 "freshness": {
 "last_successful_run_utc": "2026-07-05T09:12:00Z",
 "now_utc": "2026-07-05T15:40:00Z",
 "staleness_minutes": 388,
 "status": "BREACHED"
 },
 "upstream": [
 {"model": "staging.stg_orders", "last_run_utc": "2026-07-05T15:05:00Z", "status": "success"},
 {"model": "staging.stg_customers", "last_run_utc": "2026-07-05T14:58:00Z", "status": "success"},
 {"model": "intermediate.int_order_enriched", "last_run_utc": "2026-07-05T09:12:00Z", "status": "failed", "error":
"DBT_RUN_TIMEOUT after 45m"}
],
 "downstream_consumers": ["Looker Revenue Overview", "exec_daily_orders Slack bot"],
 "note": "int_order_enriched has been failing since morning; fct_orders has not refreshed."
}
```
```

Agent response

6:56 AM - Responded in 18s

TL;DR

- Blocking upstream: `intermediate.int\_order\_enriched` (failed with a DBT run timeout at 09:12 UTC)
- Blast radius downstream: `Looker Revenue Overview` dashboard & `exec\_daily\_orders` Slack bot - both are now serving data that is 6½h stale (388min) and any decisions/alerts based on them are out of date.
- Restore plan (ordered, with approximate "time to complete" windows): see the step by step table below. If each step is hit or miss, the whole SLA window (120min) can be recovered within the next 90min (including verification and communication).

1?? Quick Situation Assessment

| Item | Value | Comment |
|------------------------------|--|--|
| Mart | `analytics.fct_orders` | P1 revenue mart - highest priority |
| Current staleness | 388min (6h28m) | SLA breached (max 120min) |
| Last successful upstream run | `stg_orders` @ 15:05 UTC, `stg_customers` @ 14:58 UTC | both OK |
| Failed upstream | `intermediate.int_order_enriched` @ 09:12 UTC | failed (DBT_RUN_TIMEOUT after 45m) |
| Downstream consumers | - Looker Revenue Overview
 - exec_daily_orders Slack bot | Both are now delivering stale numbers to execs & board |
| Owner | `data?platform` team | Primary contact for remediation |

Root cause hypothesis - The `int\_order\_enriched` model timed out after 45min. Typical reasons:

1. Data volume spike (e.g., end of day order dump) causing a full refresh to exceed the default DBT timeout (30min).
2. Lock contention on the `orders` fact table (e.g., another long running ETL holding a lock).
3. Warehouse concurrency / credit throttling - the warehouse auto scaled down during the run, reducing available compute.

We will confirm the exact cause in Step 2.

2?? Ordered Restoration Steps

| # | Action | Owner | Why / What to Look For | Estimated Time |
|---|---|---------------------|------------------------|----------------|
| 1 | Pull the DBT run logs for `int_order_enriched` (via Airflow/Dbt Cloud UI or `dbt run --model intermediate.int_order_enriched --debug`). | Data Platform (you) | Verify the exact | |

Engineering counterpart

6:56 AM - Responded in 0s

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

Agent response

6:56 AM - Responded in 6s

? How I Got From the JSON to the Final "What?to?Do" Answer

Below is a step?by?step narrative of the mental (and practical) work I performed, with every assumption I made explicit. Feel free to interrupt at any point if you need more detail on a particular step.

1?? Read & Parse the Payload

| Field | Value | Immediate implication |
|-------------------------------|---|--|
| `mart` | `analytics.fct_orders` | The target that missed its SLA. It is a P1 revenue mart - highest business impact. |
| `sla.max_staleness_minutes` | `120` | The "freshness contract": the mart must be ??2?h old. |
| `freshness.staleness_minutes` | `388` | Current staleness = 6?h?28?m, well over the SLA ->?BREACHED. |
| `upstream` | (array) 3 models | We need to see which of these is preventing the mart from running. |
| `upstream[2]` | - `intermediate.int_order_enriched` `last_run_utc = 2026?07?05T09:12:00Z`, `status = failed`, `error = DBT_RUN_TIMEOUT after 45m` | Only one upstream model reports a failure. The other two (`stg_orders`, `stg_customers`) succeeded just minutes ago. |
| `downstream_consumers` | `Looker Revenue Overview`, `exec_daily_orders Slack bot` | Anything that reads the mart is now receiving stale data. |
| `note` | `"int_order_enriched has been failing since morning; fct_orders has not refreshed."` | Reinforces that the failure of `int_order_enriched` is the root cause of the mart's staleness. |

Assumption #1 - The mart refresh is triggered automatically after all its upstream dependencies finish successfully (standard DBT "run?all" or Airflow DAG pattern).

Assumption #2 - No manual "force?run" flag has been set; therefore the mart will wait for its upstream to succeed.

2?? Identify the Blocking Upstream Model

| Model | Last run | Status | Why it blocks |
|----------------------|-----------|------------------|---|
| `stg_orders` | 15:05?UTC | success | Already succeeded -> does not block. |
| `stg_customers` | 14:58?UTC | success | Same as above. |
| `int_order_enriched` | 09:12?UTC | failed (timeout) | Downstream mart `fct_orders` depends on this model; DBT will not materialise `fct_orders` until all upstream nodes have a successful run. |

Thus the single point of failure is `intermediate.int\_order\_enriched`.

Assumption #3 - The dependency graph is linear for this mart (i.e., `fct\_orders` <-> `int\_order\_enriched` <-> the two staging tables). If there were additional hidden dependencies they would appear in the metadata; none do.

3?? Determine the Blast?Radius (Who Is Affected?)

The JSON lists two downstream consumers. Because the mart is P1 and powers a Revenue Overview dashboard, the impact is:

| Consumer |
|----------|
|----------|

Engineering counterpart

6:56 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

6:56 AM - Responded in 19s

? "Ship?to?Prod" Verification Checklist

Before you promote any fix (code change, configuration tweak, or operational workaround) that will unblock `intermediate.int\_order\_enriched` and consequently refresh `analytics.fct\_orders`, run through the four?column checklist below.

| ? What to Verify | Why it Matters (Risk Mitigated) | How to Verify (concrete actions) | Owner / Tool |

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

| 1. Re?runability of the upstream model
??`intermediate.int\_order\_enriched` runs to completion without timing?out. | Guarantees the blockage is truly cleared; avoids a repeat breach. | - Execute a single?model DBT run (or Airflow task) with `--full-refresh` in a staging environment.
- Capture the run log and confirm status = success, run_time < timeout (default 30?min, or your custom limit).
- If you had to raise the timeout, record the new value and why. | Data?Platform Engineer (you) - DBT/airflow UI, `dbt run --model intermediate.int\_order\_enriched --debug` |

| 2. Data?quality sanity checks on the refreshed model | Prevents corrupted or partially?loaded rows from contaminating downstream revenue reports. | - Spot?check row?counts vs. yesterday's successful run (`SELECT COUNT()`).
- Run any existing dbt tests (unique, not?null, referential integrity) - they should pass.
- Compare a few key aggregates (e.g., total order amount, order?count by day) against the source system (the raw `orders` table) for the same time window. | Data?Quality Engineer - dbt test suite, custom SQL scripts |

| 3. Performance / resource consumption | A model that now runs successfully but exhausts warehouse credits or creates lock contention will cause downstream failures later. | - Look at the warehouse query profile (Snowflake UI, Redshift STL, BigQuery jobs). Capture duration, bytes scanned, concurrency slots used.
- Verify that the run stays under the configured timeout and under the resource?quota you have for the P1 pipeline (e.g., ???WH clusters).
- If you had to increase the warehouse size, ensure the cost impact is documented and approved. | Cloud?Warehouse Ops - Snowflake UI -> History -> Query Details |

| 4. Dependency graph integrity | A hidden downstream edge could be inadvertently broken (e.g., a new column that downstream models expect). | - Regenerate the dbt DAG (`dbt ls --select +intermediate.int\_order\_enriched`) and confirm that `analytics.fct\_orders` is downstream.
- Run `dbt source freshness` for any source tables that feed `int\_order\_enriched` to make sure they are current. | Data?Platform Engineer - dbt CLI |

| 5. Integration?level sanity (Downstream consumers) | Even if the model looks good, the Looker view or Slack bot could still be showing stale data due to caching or materialisation delays. | - After the mart refreshes, manually open the Looker Revenue Overview dashboard and verify the last refreshed timestamp in the UI (or via Looker API).

Engineering counterpart

6:56 AM - Responded in 0s

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

Agent response

6:56 AM - Responded in 19s

? Top Risks When Unblocking `int\_order\_enriched`

| # | Risk | Likelihood | Impact (to business) | Why it can happen | Mitigation (pre-run) |

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

| 1?? Timeout / runaway query | The model may finish once but still hit the warehouse timeout again (or exceed credit limits). | Medium - we already saw a 45min timeout. | Freshness SLA remains breached; downstream dashboards stay stale; possible cascade to other marts that share the same source tables. | - Pre-run the model with `--timeout 90m` and monitor query profile.
- If the query is a full-refresh, consider incremental materialisation or partition pruning. |

| 2?? Data quality degradation | A partial load (e.g., some partitions succeed, others fail) could slip through if we only check overall "success". | Low-Medium - DBT marks the run as success only if all statements succeed, but logical bugs can still produce bad rows. | Revenue numbers become inaccurate -> bad business decisions, audit issues. | - Run all dbt tests on the model (unique, not-null, FK).
- Spot-check aggregates against source OLTP table. |

| 3?? Lock / concurrency contention | `int\_order\_enriched` may be competing for locks on the large `orders` raw table with another nightly job (e.g., nightly CDC load). | Medium | The job could be killed or cause the CDC load to fail, breaking upstream pipelines. | - Check warehouse lock view (e.g., Snowflake `SHOW LOCKS`) before start.
- Schedule the run after the CDC load completes, or add a dependency in the orchestrator. |

| 4?? Cost spike | Extending the timeout or scaling the warehouse up can cause an unexpected jump in cloud spend. | Low-Medium | Budget overruns; finance team alerts. | - Capture credits used in the profiling step; set an alert threshold (e.g., >1.5x usual). |

| 5?? Downstream cache staleness | Even after the mart refreshes, Looker or the Slack bot may still serve the old cached result for a few minutes. | Low | Users may still see stale numbers and think the fix failed. | - Issue a cache-clear command (Looker "Refresh Explore", Slack bot webhook) after the mart is marked `success`. |

| 6?? Schema drift | If `int\_order\_enriched` introduced a new column that downstream models/dashboards are not expecting, they could error out. | Low | Dashboard breaks, alerts fail. | - Run `dbt ls --select analytics.fct\_orders+` to ensure downstream models still compile; run a quick compile-only (`dbt compile`) before materialisation. |

\Likelihood is a qualitative estimate based on the symptom (timeout) and typical patterns in our environment.

? Rollback / "Undo" Plan (If the fix doesn't bring the mart back within SLA)

| Phase | Action | Command / UI step | Owner | Success Criteria |

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

| A. Immediate stop | Abort the running DBT job (or Airflow task) for `int\_order\_enriched`. | Airflow -> Task Instance -> "Mark Failed" / DBT Cloud -> Cancel Run | Data Platform Engineer | No further compute is being consumed; the DAG is in a clean state. |

|

Engineering counterpart

6:56 AM - Responded in 0s

Thanks - upstream blocker and blast radius are well ordered. Prioritize revenue-facing dashboards if full freshness slips past the hour mark. Review complete - no further response needed.