



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-4.1-mini

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

Duration: 39s

Actual run cost: \$0.004640

Projected execution cost: \$5.10/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



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: 9756 chars - Latency: 39s - 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: 39s

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

6:55 AM - Responded in 13s

Freshness SLA Breach Triage Report

1. Blocker Upstream Model

- Model: `intermediate.int\_order\_enriched`
- Last Run: 2026-07-05T09:12:00Z
- Status: Failed (Error: DBT_RUN_TIMEOUT after 45m)
- Impact: This failure is blocking the downstream mart `analytics.fct\_orders` from refreshing. The last successful run of the mart matches the last run of this upstream model, indicating a direct dependency.

2. Blast Radius on Downstream Consumers

- Affected Consumers:
 - Looker Revenue Overview (P1 revenue dashboard)
 - exec_daily_orders Slack bot
- Impact:
 - Both consumers rely on `analytics.fct\_orders`, which is stale by 388 minutes (>120 min SLA).
 - Revenue reporting and executive alerts are potentially showing outdated data, risking decision-making based on stale information.

3. Ordered Steps to Restore the Mart within SLA

1. Investigate and Fix `int\_order\_enriched` Failure
 - Review detailed logs for the DBT run timeout cause.
 - Check warehouse performance and resource availability.
 - Optimize long-running queries or break the job into smaller steps.
 - If possible, rerun the failing model with higher timeout settings or increased compute resources.
2. Rerun `intermediate.int\_order\_enriched`
 - Ensure it completes successfully.
 - Monitor closely to confirm no further timeouts.
3. Trigger Refresh of `analytics.fct\_orders`
 - Once `int\_order\_enriched` is successful, run or schedule the mart refresh.
 - Confirm the mart completes within the SLA.
4. Validate Data Freshness and Notify Stakeholders
 - Verify the data freshness timestamp meets the SLA.
 - Inform the owner team (`data-platform`) and downstream consumer teams that the issue is resolved.
5. Postmortem and Prevention
 - Analyze root cause for timeout (e.g., data volume, query inefficiency).
 - Implement alerting for upstream model failures earlier.
 - Consider pipeline adjustments to prevent cascading SLA breaches.

Summary

- Root Cause: `intermediate.int\_order\_enriched` failing due to DBT_RUN_TIMEOUT.
- Blocked Mart: `analytics.fct\_orders` stale for 388 minutes.
- Impacted Dashboards: Looker Revenue Overview and exec_daily_orders Slack bot.
- Next Actions: Fix and rerun upstream model, then refresh mart.

Please reach out if you need assistance with logs or cluster resource allocation.

Engineering counterpart
6:55 AM - Responded in 0s

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

Agent response

6:55 AM - Responded in 11s

Certainly! Here's a step-by-step walkthrough of my reasoning, along with the assumptions I made:

Step 1: Understand the SLA Breach Context

- The mart `analytics.fct\_orders` breached its freshness SLA:
 - SLA max staleness: 120 minutes
 - Actual staleness: 388 minutes (significantly over the limit)
- This indicates the mart has not successfully refreshed for over 6 hours.

Step 2: Identify the Cause by Examining Upstream Model Statuses

- Looked at the `upstream` array:
 - `staging.stg\_orders`: last success at 15:05 UTC (very recent)
 - `staging.stg\_customers`: last success at 14:58 UTC (very recent)
 - `intermediate.int\_order\_enriched`: last run at 09:12 UTC, but failed with a `DBT\_RUN\_TIMEOUT` after 45m`.
- The mart's last successful run matches the last run of `int\_order\_enriched` (09:12 UTC), indicating it relies directly on `int\_order\_enriched`.
- Assumption: The mart depends on `int\_order\_enriched` and cannot refresh until it succeeds.

Step 3: Analyze the Failure Impact and Blast Radius

- Given `int\_order\_enriched` has been failing since morning, the mart refreshes are blocked.
- Downstream consumers:
 - Looker Revenue Overview (P1 revenue dashboard, hence critical)
 - exec_daily_orders Slack bot (likely used for executive alerts)
- Assumption: These consumers rely on up-to-date `analytics.fct\_orders` data.
- Impact is moderate to high because the dashboard is P1-tier and delayed by over 6 hours.

Step 4: Develop a Recovery Plan

- Fixing the root cause-to get `int\_order\_enriched` to run successfully-is priority.
- Timeout indicates a long-running or resource-heavy job.
- Steps:
 1. Debug and fix the failing model (review logs, resource issues).
 2. Rerun `int\_order\_enriched`.
 3. Once successful, rerun `analytics.fct\_orders`.
 4. Confirm freshness SLA is met.
 5. Notify stakeholders.
- Assumption: Once the upstream model runs successfully, the mart will refresh without issues.
- The ability to re-trigger the mart quickly is assumed.

Step 5: Consider Preventative Actions

- Timeout could be due to inefficient queries, large data volume, or cluster constraints.
- Recommend postmortem and alerting improvements to catch issues earlier.

Summary of Assumptions Made

- Dependency: `analytics.fct\_orders` depends directly on `intermediate.int\_order\_enriched`.
- Mart refresh depends on successful upstream runs.
- The reported error (`DBT\_RUN\_TIMEOUT`) indicates resource or query performance issues.
- Downstream consumers read the mart directly and rely on its freshness.
- The environment allows manual or automated reruns of models and marts.
- The owner team (data-platform) is responsible for fixing the upstream failure.

If you want, I can also help draft next steps for debugging the timeout or suggest query optimization.

Engineering counterpart
6:55 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

6:55 AM - Responded in 8s

Before shipping the triage report and remediation plan to production stakeholders, I would verify the following:

1. Accuracy of Dependency Graph

- Confirm that `analytics.fct\_orders` indeed depends on `intermediate.int\_order\_enriched` and that no other upstream models are blocking.
- Verify no hidden dependencies or downstream freshness blockers.

2. Freshness and Status Timestamps

- Double-check timestamps and timezone consistency to ensure staleness and last run times are correct.
- Confirm that the failure timestamp and freshness breach timeline align logically.

3. Failure Details

- Validate the exact error cause for the `DBT\_RUN\_TIMEOUT` failure:
 - Confirm it is not a transient infrastructure issue.
 - Check if there are detailed logs or stack traces that might point to query bottlenecks or external system failures.

4. Impact on Downstream Consumers

- Confirm that the reported downstream consumers (`Looker Revenue Overview` and `exec\_daily\_orders Slack bot`) are actively using the stale mart.
- Assess if there are additional consumers or reports impacted.

5. Environment and Recovery Capabilities

- Validate ability to safely rerun the failing model and mart without causing cascading failures or data inconsistencies.
- Verify that rerun permissions, compute resources, and time windows exist to meet SLA after remediation.

6. Communication and Escalation Paths

- Confirm the owner team (`data-platform`) is aware and ready to act.
- Make sure key stakeholders for downstream consumers are informed or can be promptly notified.

7. Preliminary Mitigation Checks

- Check if any quick mitigation (e.g., increasing timeout or resources, partial runs) can be applied to restore freshness faster.
- Confirm no partial data or inconsistent state will persist after reruns.

Summary

Verifying these points ensures that the triage is accurate, the proposed remediation is safe, and the communication plan considers all affected parties. This reduces risk of misinformation, repeated failures, or missed impact before escalating or executing the fix.

Engineering counterpart
6:55 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 8s

Top Risks

1. Persistent Timeout on `int\_order\_enriched`
 - The DBT run could continue to time out due to query complexity, data volume growth, or warehouse resource limits.
 - This would prolong the mart staleness and SLA breach.
2. Cascading Failures or Data Inconsistency
 - If reruns are forced without resolving root causes, partial or inconsistent data could propagate downstream.
 - This might lead to incorrect reporting or executive decisions.
3. Insufficient Compute Resources
 - Warehouse cluster limits or concurrency issues could block or slow model runs.
 - Rerunning may fail again or impact other workloads.
4. Undocumented Dependencies or Consumers
 - Other downstream systems might be impacted but currently unknown, increasing blast radius.
 - Communication gaps could cause surprise outages.
5. Delay in Stakeholder Notification
 - If stakeholders are not informed timely, business decisions could be made on stale data unknowingly.

Rollback Plan

If remediation attempts fail or worsen the situation:

1. Pause Automated Refreshes
 - Temporarily halt scheduled refreshes of both `int\_order\_enriched` and `analytics.fct\_orders` to prevent repeated failures or resource exhaustion.
2. Revert to Last Known Good State
 - Ensure the mart remains at the last successful refresh state (as of 09:12 UTC).
 - Avoid partial updates until root cause is fixed.
3. Notify Stakeholders Immediately
 - Communicate the ongoing issue, impact, and expected timelines.
 - Provide transparency to downstream consumer teams and executives.
4. Engage Engineering Support
 - Escalate to DBT experts, warehouse admins, or platform engineers to deep-dive into query optimization, resource scaling, or potential bugs.
5. Implement Temporary Workarounds
 - Consider using cached or interim datasets for critical dashboards.
 - Explore data partition skipping or incremental runs if supported.
6. Schedule Postmortem and Preventive Measures
 - Once stable, conduct a thorough root cause analysis and improve monitoring/alerting to avoid recurrence.

This rollback plan focuses on minimizing further disruption, maintaining data consistency, and ensuring clear

communication while a sustainable fix is identified.

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.