



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: sao10k/l3-lunaris-8b

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

Duration: 28s

Actual run cost: \$0.000310

Projected execution cost: \$0.3410/mo (50/biz day × 22 days)

Report date: July 14, 2026 - 2:57 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: 10435 chars - Latency: 28s - 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: 28s

Engineering counterpart

6:57 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

6:57 AM - Responded in 6s

Summary:

The analytics.fct_orders mart, a P1 revenue dashboard, is stale by 388 minutes and has breached its freshness SLA. The root cause is the failed refresh of the intermediate.int_order_enriched model 8 hours ago due to a DBT_RUN_TIMEOUT. This failure is blocking the refresh of the fct_orders mart, affecting 2 downstream consumers.

Blast Radius:

The failure of intermediate.int_order_enriched has cascading impacts:

1. fct_orders mart has not refreshed since the enrichment failed 8 hours ago.
2. The Looker Revenue Overview, which relies on fct_orders for data, is also impacted and out of date.
3. The exec_daily_orders Slack bot, another consumer of fct_orders, will not produce accurate reports until the mart is refreshed.

Restoration Steps:

1. Quick fix: Run the failed intermediate.int_order_enriched model as soon as possible to make fct_orders refreshable. This will bridge the staleness temporarily until a more permanent fix is implemented.
2. Root cause analysis and resolution: Investigate the DBT_RUN_TIMEOUT issue in int_order_enriched, which was caused by a 45-minute timeout. This may require:
 - Increasing the timeout in dbt_project.yml or the specific model file.
 - Optimizing the query or data processing in the model to complete within the current timeout.
 - Implementing circuit breakers or retries to handle occasional timeouts.
3. Monitoring and maintenance: Regularly review the execution and success of the int_order_enriched model, ensuring it completes within a reasonable timeframe. Automate monitoring and alerting to detect potential future failures.
4. SLA mitigation: Consider implementing a warm standby or materialized view of fct_orders to provide a more immediate response to future failures of int_order_enriched, reducing the blast radius and staleness.

By completing these steps, we can restore the freshness of analytics.fct_orders within the SLA and mitigate the impact on downstream consumers.

Engineering counterpart

6:57 AM - Responded in 0s

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

Agent response

6:57 AM - Responded in 9s

Let's break down the analysis step by step:

1. Identifying the root cause:

The freshness SLA breach is triggered by the staleness of the `fct_orders` mart, with 388 minutes of staleness. The upstream models are relevant to understanding why `fct_orders` didn't refresh. The failed `intermediate.int_order_enriched` model stands out as the most likely culprit, given its "failed" status and error message.

Assumption: The `fct_orders` mart refreshes by processing the latest data from its immediate upstream models, specifically `staging.stg_orders` and `staging.stg_customers`, and then enriching that data in `intermediate.int_order_enriched` before populating the fact table.

2. Understand the impact of the failure:

The `int_order_enriched` model has been failing for 8 hours, which means `fct_orders` has not refreshed since then. This staleness is cascading to the mart's downstream consumers.

Assumption: The downstream consumers (Looker Revenue Overview and `exec_daily_orders` Slack bot) rely directly on the `analytics.fct_orders` mart for their data, and their freshness is tied to the mart's refresh cycle.

3. Temporary solution (quick fix):

To address the immediate need for `fct_orders` to be refreshable, running the failed `int_order_enriched` model now will allow the fact table to be populated and bring the mart's staleness within the SLA window.

Assumption: The temporary fix will not address the underlying issue causing the `int_order_enriched` failure but will stabilize the situation until a more permanent solution is implemented.

4. Resolution and avoidance of future failures:

The real issue is the 45-minute timeout in `int_order_enriched`, which is causing the model to fail. To prevent future failures and ensure timely refreshes:

- Check if the timeout can be adjusted safely in the model configuration.
- If the timeout is reasonable, investigate optimizing the query performance or processing to complete within that timeframe.
- Consider implementing circuit breakers or retries to handle occasional timeouts without causing cascading failures.

Assumption: The optimal solution will depend on the specifics of the `int_order_enriched` model, its data volumes, and the underlying system performance.

5. SLA mitigation and monitoring:

To further minimize the risk of future breaches:

- Implement monitoring to proactively detect `int_order_enriched` failures and alert the team.
- Consider strategies like materialized views or warm standbys to provide a more immediate response to failures, reducing the staleness and blast radius.

Assumption: These SLA mitigation strategies will require additional development, testing, and infrastructure setup.

Throughout the analysis, I made the following key assumptions:

1. The mart's refresh workflow and dependencies are as described.
2. The primary issue is the failed `int_order_enriched` model affecting `fct_orders` staleness.
3. The temporary fix and resolution will resolve the immediate SLA breach and prevent future similar incidents.
4. The proposed SLA mitigation strategies are feasible and worth implementing to enhance system robustness.

Please let me know if you have any further questions or if you'd like me to elaborate on any part of the analysis!

Engineering counterpart
6:57 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

6:57 AM - Responded in 6s

Before implementing these changes in production, I would verify the following:

1. Correctness:

- Ensure the temporary fix for fct_orders refreshes correctly and within the SLA window.
- Verify the resolution for int_order_enriched successfully addresses the DBT_RUN_TIMEOUT issue.
- Confirm that the mart's data accuracy and integrity have not been compromised by the temporary fix or resolution.

2. Thorough testing:

- Perform a suite of tests, including unit tests, integration tests, and end-to-end simulations, to validate the correctness and robustness of the changes.
- Stress test the models and workflows to ensure they handle increased load and data volumes without failing or causing staleness.

3. Performance and resource utilization:

- Monitor the memory, CPU, and disk usage of the models and databases to ensure the changes do not introduce performance bottlenecks or lead to resource exhaustion.
- Verify the adjusted timeouts do not excessively prolong model runtimes, impacting other workflows or SLAs.

4. Monitoring and alerting:

- Set up and test the monitoring and alerting systems to detect potential future failures of int_order_enriched and other critical models.
- Ensure the alerts are actionable, clearly indicating the nature of the issue and the necessary corrective actions.

5. Backup and recovery:

- Verify that backup processes are in place to capture data changes during the fix and resolution implementation.
- Validate the recovery process to ensure that, in the event of a failure during the update, the system can be rolled back to a known good state without data loss.

6. Documentation and handover:

- Update documentation to reflect the changes made, including configuration changes, testing procedures, and any new monitoring or alerting processes.
- Ensure that the data platform team is aware of the fixes and can maintain and troubleshoot these changes if needed in the future.

7. Risk assessment and communication:

- Evaluate the potential risks associated with these changes, such as data inconsistencies, performance impacts, or unintended consequences on dependent systems.
- Communicate the changes, their benefits, and the associated risks to relevant stakeholders, including data consumers and infrastructure teams.

By thoroughly verifying these aspects, you can ensure that the changes are reliable, performant, and well-managed before deploying them to production.

Engineering counterpart
6:57 AM - Responded in 0s

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

Agent response

6:57 AM - Responded in 7s

Top risks associated with this change:

1. Data inconsistencies: The temporary fix for `fct_orders` might incorrectly process data if the `int_order_enriched` model is still experiencing issues. This could lead to data inconsistencies in the fact table.
2. Model instability: The revised `int_order_enriched` model may introduce new performance or stability problems, causing future failures or slow refreshes.
3. Overlooked side effects: The changes could have unintended consequences on other models, dependencies, or downstream consumers that rely on the refreshed `fct_orders` mart.
4. Increased latency: Optimizing the `int_order_enriched` model might reduce its processing time, but it could also increase the latency between when data is available in the staging layers and when it's propagated to the analytics mart, potentially affecting real-time reporting needs.

Rollback plan:

1. Immediate rollback: If the fixes introduce significant issues, such as data corruption or model instability, quickly revert the changes to the pre-fix state. This can usually be done by rolling back the model commits and reverting the configuration changes.
2. Gradual rollback: If the issues are more subtle or have a limited impact, consider a gradual rollback:
 - Isolate the affected models and prevent them from refreshing `fct_orders` or impacting downstream consumers.
 - Investigate and fix the root cause of the `int_order_enriched` failure without introducing the original timeout change.
 - Revert the configuration changes and reapply the original timeouts in `int_order_enriched`.
 - Monitor the system closely to ensure the rollback is successful and does not introduce new issues.
3. Communication and incident management: Throughout the rollback process, maintain open communication with stakeholders to keep them informed of the situation, the planned actions, and any changes or updates. Ensure that incident management procedures are followed, and the necessary support teams are alerted and engaged as needed.
4. Post-mortem analysis: After the situation has been resolved, conduct a thorough post-mortem analysis to understand the root causes of the issues, identify contributing factors, and document lessons learned. This will help inform future development, testing, and change management practices to minimize the risk of similar incidents.

The specific details of the rollback plan will depend on the complexity of the changes, the severity of the issues introduced, and the organizations' incident management processes. However, the key is to move quickly and carefully, maintain transparency, and have a solid understanding of the system before making changes.

Engineering counterpart

6:57 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.