



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-4o-mini

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

Duration: 31s

Actual run cost: \$0.001740

Projected execution cost: \$1.91/mo (50/biz day × 22 days)

Report date: July 13, 2026 - 10:37 PM EST

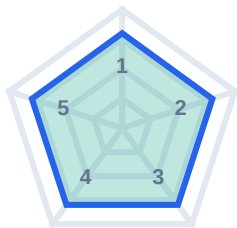
Overall

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: 13418 chars - Latency: 31s - 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: 31s

Engineering counterpart

2:37 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

2:37 AM - Responded in 6s

Triage of Freshness SLA Breach

Upstream Model Blocking Refresh:

The upstream model blocking the refresh of the `analytics.fct\_orders` mart is `intermediate.int\_order\_enriched`. This model has a status of "failed" with a recorded error of "DBT_RUN_TIMEOUT after 45m", and it has not successfully run since `2026-07-05T09:12:00Z`.

Blast Radius on Downstream Consumers:

The downstream consumers affected by the freshness SLA breach include:

1. Looker Revenue Overview - This dashboard relies on `fct\_orders` for its reporting.
2. exec_daily_orders Slack bot - This bot will provide stale order data as it depends on `fct\_orders`.

Given that `fct\_orders` is a P1 mart linked directly to revenue reporting, the impact is significant as it affects key performance indicators for the organization.

Steps to Restore the Mart within SLA

1. Investigate and Resolve the Upstream Failure:

- Check the logs for `intermediate.int\_order\_enriched` to diagnose the cause of the timeout. Investigate whether the failure is due to resource constraints, data issues, or configuration.
- If required, optimize the model or data sources to prevent future timeouts.

2. Manual Trigger of `intermediate.int\_order\_enriched`:

- Once the issue is diagnosed and resolved, manually trigger a refresh of the `intermediate.int\_order\_enriched` model to ensure it runs successfully.
- Monitor the process closely for completion without further errors.

3. Refresh the `analytics.fct\_orders` Mart:

- After `intermediate.int\_order\_enriched` has successfully refreshed, trigger a refresh of the `analytics.fct\_orders` mart.
- Validate that the refresh completes successfully and meets the freshness SLA.

4. Notify Stakeholders:

- Communicate with relevant teams (e.g., data-platform team, users of Looker Revenue Overview, and exec_daily_orders Slack bot) about the resolution of the issue, the state of the data, and the expected impact on reporting.

5. Post-Mortem Analysis:

- Conduct a post-mortem analysis to identify the root cause of the failure and implement preventive measures, such as increasing timeout settings, optimizing queries, or improving resource allocation.

6. Monitor Future Runs:

- Set up alerts to monitor the freshness of both `int\_order\_enriched` and `fct\_orders` to catch any future issues promptly before they breach SLA again.

By following these ordered steps, the freshness SLA can be restored and maintained effectively.

Engineering counterpart

2:37 AM - Responded in 0s

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

Agent response

2:37 AM - Responded in 9s

Certainly! Here's a breakdown of the reasoning I used to analyze the freshness SLA breach, along with the assumptions made at each step:

Step-by-Step Reasoning:

1. Understanding the Freshness SLA Breach:

- The metadata indicates that the `analytics.fct\_orders` mart has a maximum allowed staleness of 120 minutes. It has a recorded staleness of 388 minutes, meaning it is significantly overdue for a refresh.
- Assumption: The SLA breach is solely due to the failure of the upstream models affecting the `fct\_orders` mart, not due to any other external factors or issues unrelated to data processing.

2. Identifying the Upstream Model Blocking Refresh:

- The upstream models listed are:
 - `staging.stg\_orders` (last run successful at 2026-07-05T15:05:00Z)
 - `staging.stg\_customers` (last run successful at 2026-07-05T14:58:00Z)
 - `intermediate.int\_order\_enriched` (last run failed at 2026-07-05T09:12:00Z)
- The last successful run of `intermediate.int\_order\_enriched` is the oldest among the upstream models, indicating that it has not provided any updated data to the downstream mart since its failure.
- Assumption: I assumed that the model `int\_order\_enriched` is crucial for the data flow and that its failure directly impacts the ability of `fct\_orders` to refresh.

3. Assessing the Blast Radius on Downstream Consumers:

- The downstream consumers identified are the "Looker Revenue Overview" and "exec_daily_orders Slack bot." Both depend on up-to-date data from `fct\_orders`.
- Assumption: I assumed that these consumers need fresh data for regular operations, and the stale data may affect decision-making or reporting reliability.

4. Restoration Steps:

- Investigate and Resolve Upstream Failure: This involves checking logs for the root cause of the failure and optimizing performance. I assumed that immediate diagnostics and resolution are necessary before any subsequent actions can be taken.
- Manual Trigger of Refresh: Once the issue is resolved, a manual kick-off of the data model refresh is assumed to be the best way to expedite the fix.
- Refresh `fct\_orders`: I assumed that once `int\_order\_enriched` runs successfully, the downstream mart can be refreshed, and that this dependency needs to be honored.
- Notify Stakeholders: Keeping stakeholders informed is crucial during an SLA breach, so I assumed that timely communication would be important for transparency and operational clarity.
- Post-Mortem Analysis: This step is assumed as a standard practice to prevent recurrence of the issue.
- Monitor Future Runs: Assumed that proactive measures are essential to detect and address any future issues before they lead to another SLA breach.

Final Assumptions:

- I assumed that all models are functioning correctly under normal circumstances and that the failure is an isolated incident affecting only the `int\_order\_enriched` model.
- It is assumed that there are no broader system issues (like infrastructure failures) impacting the ability to refresh models.
- The team has the capability to resolve the identified failure and that there are channels for effective communication with stakeholders.

This structured reasoning and the assumptions made provide a clear path to systematically address the breach and restore compliance with the freshness SLA.

Engineering counterpart
2:37 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

2:37 AM - Responded in 8s

Before shipping any changes or processes to production, especially in the context of resolving a freshness SLA breach, there are several critical verification steps to ensure that the solution is robust and that potential issues have been addressed. Here's a list of what I would verify:

Verification Checklist

1. Root Cause Analysis:

- Confirm Resolution: Verify that the root cause of the ``intermediate.int_order_enriched`` failure has been thoroughly identified and resolved (i.e., the underlying issue that caused the timeout should not persist).
- Test Changes: If optimizations or code changes were made, conduct tests in a staging or development environment to ensure they effectively mitigate the timeout issue.

2. Successful Run of Upstream Model:

- Monitor the Latest Run: After resolving the failure, confirm that ``intermediate.int_order_enriched`` has run successfully without errors and that it completes within an acceptable time frame.
- Check Data Integrity: Validate that the output data from ``int_order_enriched`` is accurate and complete, ensuring it meets the expected schema and data quality standards.

3. Refresh of Downstream Mart:

- Trigger Refresh: Ensure ``analytics.fct_orders`` is refreshed successfully after ``int_order_enriched`` without any issues.
- Verify Freshness: Check that ``fct_orders`` now meets the freshness SLA criteria (i.e., ensure its staleness is below the maximum allowed threshold).

4. Impact on Downstream Consumers:

- Test Consumer Functionality: Verify that downstream consumers (e.g., Looker Revenue Overview and `exec_daily_orders` Slack bot) are receiving the updated data correctly, and check for any errors or anomalies in their output.
- Communicate with Users: Ensure that key stakeholders are informed and confirm that there are no further issues from their side due to the earlier SLA breach.

5. Monitoring and Alerts:

- Set Up Monitoring: Confirm that appropriate monitoring is in place for both the upstream and downstream models to catch any future failures or performance issues quickly.
- Configure Alerts: Ensure alerts are configured to notify relevant teams if any part of the data pipeline breaches predefined thresholds or fails.

6. Documentation and Training:

- Update Documentation: Ensure that any changes made to the data models, processes, or systems are well-documented, including the root cause of the failure and how it was resolved.
- Train the Team: If any new procedures or protocols have been implemented, ensure that the relevant team members are trained on these updates.

7. Rollback Plan:

- Create a Rollback Plan: Establish a clear rollback plan in case the changes cause unforeseen issues after deployment. This plan should detail steps to revert to the previous stable state.

8. Peer Review:

- Conduct a Peer Review: Have another team member or technical lead review the changes and the resolution process. This provides an opportunity for additional insights and verification of the changes before deployment.

Final Checks

- Compliance with Standards: Ensure that all changes comply with internal development and deployment standards and practices.
- Performance Testing: If applicable, conduct performance testing to confirm that the changes do not negatively impact the

overall system performance.

By going through this comprehensive verification process, we minimize the risk of introducing new issues into the production environment and ensure that the freshness SLA is restored and maintained effectively.

Engineering counterpart
2:37 AM - Responded in 0s

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

Agent response

2:37 AM - Responded in 8s

When implementing changes to resolve a freshness SLA breach, it's important to identify potential risks and prepare a rollback plan to mitigate any issues that may arise during or after deployment. Here are the top risks associated with this situation, along with a detailed rollback plan.

Top Risks:

1. Upstream Model Stability:

- Risk: The changes made to resolve the timeout in `intermediate.int_order_enriched` could introduce new bugs or instability, resulting in subsequent failures.
- Mitigation: Thorough testing in a staging environment and monitoring after deployment will help catch issues early.

2. Downstream Data Integrity:

- Risk: The refreshed `analytics.fct_orders` mart may contain incorrect or incomplete data, affecting the reports and dashboards that rely on it.
- Mitigation: Validate data integrity immediately after the refresh through data quality checks and comparisons with expected outputs.

3. Performance Issues:

- Risk: Modifications to the upstream model may negatively affect processing times, leading to performance degradation in the data pipeline.
- Mitigation: Conduct performance testing and monitor system resources closely after deployment.

4. User Impact:

- Risk: End-users of the downstream consumers may experience disruptions or discrepancies in reporting while changes are being implemented or if something goes wrong.
- Mitigation: Communicate with users about possible disruptions and set expectations regarding data availability.

5. Monitoring Failures:

- Risk: Monitoring and alerting systems may not function properly, leading to undetected failures in the pipeline.
- Mitigation: Test monitoring tools and ensure that alerts are configured correctly before deployment.

Rollback Plan:

If the changes do not work as anticipated, the following rollback plan can be executed:

1. Immediate Notification:

- Alert the relevant stakeholders and users about the issue, informing them that a rollback will be initiated to restore normal operations.

2. Stop Any Ongoing Processes:

- Halt any data processing jobs related to the upstream model and the `fct_orders` refresh to prevent further complications or data corruption.

3. Revert Code Changes:

- Restore the previous version of the code for `intermediate.int_order_enriched` (and any other affected models) from the version control system (e.g., Git). This should be the last stable version that did not encounter timeouts.

4. Re-Deploy Previous Stable Version:

- Redeploy the last known stable version of the models to ensure that the system returns to a functional state quickly.

5. Re-run Successful Processes:

- If changes were made that affected models, re-trigger the last successful runs of all relevant upstream models to ensure the data is consistent and up to date.

6. Validate Data:

- After rolling back, validate the data in ``analytics.fct_orders`` to ensure it reflects the last consistent state prior to the changes.

7. Re-assess Monitoring and Alerts:

- Review and ensure that monitoring and alerting setups are functioning correctly to prevent similar issues from going unnoticed in the future.

8. Document the Rollback:

- Document the rollback process, including the reasons for the rollback and any performance insights, to help inform future deployments.

9. Conduct a Post-Rollback Review:

- Gather the team for a post-mortem review to discuss what went wrong, analyze the effectiveness of the rollback, and determine steps for a safer re-implementation of the changes.

By proactively identifying risks and having a detailed rollback plan, we can effectively manage potential issues and safeguard the integrity of the data pipeline and the overall data quality for our users.

Engineering counterpart**2:37 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.