



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: mistralai/mistral-nemo

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

Duration: 2m 46s

Actual run cost: \$0.000156

Projected execution cost: \$0.1716/mo (50/biz day × 22 days)

Report date: July 13, 2026 - 10:38 PM EST

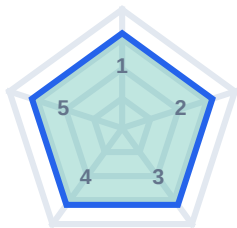
Overall

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: 12120 chars - Latency: 166s - 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: 2m 46s

Engineering counterpart

2:36 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

2:36 AM - Responded in 47s

Analysis:

- The breach status was triggered due to the `fct\_orders` mart not refreshing within the SLA of 120 minutes. The mart has been stale for 388 minutes.
- Upstream blocker: The `intermediate.int\_order\_enriched` model failed due to a `DBT\_RUN\_TIMEOUT` error after 45 minutes. This model is blocking the refresh of `analytics.fct\_orders`.
- Downstream consumers affected by this breach are:
 - Looker's "Revenue Overview" dashboard, which is likely showing stale data.
 - "exec_daily_orders" Slack bot, which may be sending inaccurate or outdated information.

Steps to restore the mart within the SLA:

1. Troubleshoot and resolve the upstream issue:

- Investigate the `DBT\_RUN\_TIMEOUT` error for the `int\_order\_enriched` model.
- Check if there are any known issues with the tasks that `int\_order\_enriched` depends on.
- Look into the model's logs for detailed error messages to understand the cause of the timeout.
- If necessary, temporarily disable or exclude resources consuming heavy computation or disk I/O in the affected model to expedite its execution.

2. Retry the failed model:

- Once the issue is understood and addressed, retry running the `int\_order\_enriched` model.
- Monitor the run to ensure it completes within the expected timeframe.

3. Retry the affected mart:

- After the `int\_order\_enriched` model completes successfully, retry running the `fct\_orders` mart.
- Monitor the run to ensure it completes within the expected timeframe.

4. Verify data freshness and accuracy:

- Confirm that the `fct\_orders` mart has been updated with fresh data.
- Perform a quick data validation to ensure no data integrity issues have occurred during the restoration process.

5. Notify downstream consumers:

- Inform the teams using the "Revenue Overview" dashboard and the "exec_daily_orders" Slack bot that the freshness issue has been resolved.

6. Monitor and follow up:

- Keep an eye on the affected mart to ensure it continues to refresh within the SLA.
- Follow up on the error that caused the timeout to prevent it from recurring in the future.

Engineering counterpart

2:36 AM - Responded in 0s

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

Agent response

2:38 AM - Responded in 1m 12s

Step-by-step reasoning and assumptions:

1. Identify the upstream blocker:

- The metadata JSON states that the `fct\_orders` mart has not refreshed since "2026-07-05T09:12:00Z", and the last successful run of its upstream `int\_order\_enriched` model is at the same time. This indicates that the failure of `int\_order\_enriched` is preventing `fct\_orders` from refreshing.
- The `int\_order\_enriched` model failed due to a `DBT\_RUN\_TIMEOUT` error, suggesting that the model took too long to execute and was stopped by the system. This is a clear indication that something is wrong and needs to be addressed before proceeding.
- The other upstream models (`stg\_orders` and `stg\_customers`) had successful runs recently, so they are not blocking the refresh of `fct\_orders`.

Assumptions made:

- The dependency hierarchy is correctly defined, and the `fct\_orders` mart depends on the `int\_order\_enriched` model.
- The `DBT\_RUN\_TIMEOUT` error is the sole reason preventing `fct\_orders` from refreshing.

2. Determine downstream impact:

- The metadata JSON lists "Looker Revenue Overview" and "exec_daily_orders Slack bot" as downstream consumers. These consumers rely on fresh data from the `fct\_orders` mart.
- Since the mart has been stale for 388 minutes, both consumers have likely been using outdated data during this period.

Assumptions made:

- The listed downstream consumers are the only ones affected by the freshness breach.
- The consumers have not implemented any fallbacks or workarounds to handle freshness issues.

3. Plan the restoration steps:

- To restore the mart within the SLA, the upstream issue causing the timeout needs to be addressed first.
- Once the issue is resolved, retrying the failed model and then the affected mart should bring fresh data back to the downstream consumers.
- The restoration steps follow a logical dependency chain: resolve upstream blocker -> retry failed model -> retry affected mart -> verify freshness and accuracy -> notify downstream consumers.

Assumptions made:

- Addressing the timeout issue will resolve the failure of the `int\_order\_enriched` model.
- Retrying the model and the mart will succeed once the upstream issue is resolved.
- Notifying downstream consumers after resolving the issue is sufficient to alert them about the restored freshness.

4. Prioritize actions:

- Troubleshooting and resolving the upstream timeout issue should be prioritized, as it is the root cause preventing the mart from refreshing.
- Once the upstream issue is addressed, quickly retrying the failed model and the affected mart is crucial to minimize the breach duration.
- Verifying data freshness and accuracy, as well as notifying downstream consumers, are essential steps to ensure that the issue has been effectively resolved.

Assumptions made:

- Addressing the upstream issue in a timely manner will result in the shortest recovery time.
- Quickly retrying the affected models and verifying freshness will minimize the overall breach duration.

By following this step-by-step reasoning process and making the listed assumptions, you can effectively approach the freshness SLA breach and create a plan to restore the mart within the specified service level agreement.

What would you verify before shipping this to production?

Agent response

2:38 AM - Responded in 19s

Before shipping the restored mart to production, it's essential to verify several aspects to ensure data accuracy, stability, and minimal impact on downstream consumers. Here's a list of verifications to perform:

1. Complete the troubleshooting and resolution process:
 - Ensure that the root cause of the timeout issue has been identified and addressed.
 - Verify that the ``int_order_enriched`` model runs successfully without encountering the ``DBT_RUN_TIMEOUT`` error again.
2. Successfully run the affected mart:
 - Retry the ``fct_orders`` mart after the upstream model has been resolved.
 - Monitor the run to ensure it completes within the expected timeframe and without any errors.
3. Data validation and accuracy checks:
 - Compare the data in the restored mart with the previous successful run or a known good state to ensure data consistency and accuracy.
 - Perform basic data validation, such as checking for null values, data types, and expected count ranges.
 - Verify that the restored mart contains fresh data covering the expected time range.
4. Downstream impact and consumer verification:
 - Access the downstream consumers (e.g., Looker dashboard, Slack bot) to confirm that they now show fresh data from the restored mart.
 - Check if the downstream consumers have any error messages or warnings related to the freshness breach.
 - Ensure that the restored mart's data aligns with the expectations of the downstream consumers.
5. Performance and resource utilization:
 - Monitor the performance of the restored mart and its upstream models during the initial runs to ensure they meet the desired performance expectations.
 - Verify that the restored mart and its upstream models do not cause excessive resource utilization, which could impact other workloads in the environment.
6. Retry the SLA breach scenario (if possible):
 - If the root cause of the timeout issue allows, attempt to reproduce the SLA breach scenario under controlled conditions to validate that the applied fix resolves the issue consistently.
 - This step helps ensure that the system can handle potential future SLA breaches gracefully.
7. Update monitoring and alerting:
 - Review and update the monitoring and alerting setup for the affected mart to ensure it can detect and alert on freshness breaches promptly.
 - Ensure that the alerting system is capable of handling multiple consecutive failures or slow-running models to avoid alert fatigue.
8. Document the incident:
 - Document the incident, including the root cause, troubleshooting steps, and the resolution process.
 - Update relevant knowledge bases and onboarding materials to ensure that others can learn from the incident and apply the lessons learned.

By verifying these aspects, you can minimize the risk of introducing new issues to production and ensure that the restored mart meets the required data freshness, accuracy, and performance expectations.

Engineering counterpart
2:38 AM - Responded in 0s

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

Agent response

2:38 AM - Responded in 28s

Top risks and rollback plan:

Top risks:

1. Data inconsistencies or inaccuracies: During the troubleshooting and restoration process, data inconsistencies or inaccuracies might have been introduced, leading to incorrect downstream insights.
2. Performance degradation: Resolving the timeout issue or applying temporary workarounds may introduce performance issues, impacting the affected mart and its downstream consumers.
3. Service disruption: If the restoration process fails or introduces new issues, it may cause a service disruption, leaving downstream consumers without fresh data for an extended period.
4. Recurring SLA breaches: If the root cause of the initial timeout issue was not properly addressed or if the applied fix is temporary, recurring SLA breaches may occur, leading to further data staleness and potential loss of trust in downstream insights.
5. Impact on other workloads: Increased resource utilization during the restoration process or as a result of applied workarounds may impact other workloads, affecting their performance or causing service level agreement (SLA) breaches.

Rollback plan:

1. Monitor the initial restoration attempts:
 - Closely monitor the restoration process to detect any signs of data inconsistencies, performance issues, or service disruptions.
 - Be prepared to roll back the changes if any critical issues are identified.
2. Establish a backup plan:
 - Before proceeding with the restoration, ensure that there is a recent, reliable backup of the affected mart and its upstream models.
 - Verify that the backup can be rapidly restored if needed.
3. Define rollback criteria:
 - Establish clear criteria for initiating a rollback, such as:
 - Data inconsistencies or inaccuracies that cannot be quickly resolved.
 - Significant performance degradation affecting the restored mart or its downstream consumers.
 - Recurring SLA breaches within a short period after applying the fix.
 - Impact on other workloads that causes SLA breaches or significant performance issues.
4. Rollback steps:
 - a. Stop the affected mart and its upstream models:
 - Temporarily stop the execution of the affected mart and its upstream models to prevent further data changes.
 - b. Restore the backup:
 - Quickly restore the backup of the affected mart and its upstream models to minimize downtime.
 - Ensure that the restored data is consistent with the previous state before the freshness SLA breach.
 - c. Re-evaluate the root cause and troubleshooting steps:
 - Review the initial assessment of the root cause and the troubleshooting steps taken.
 - If necessary, engage additional resources or seek external help to resolve the issue.
 - d. Communicate the rollback:
 - Notify the relevant teams, including downstream consumers, about the rollback and the reason behind it.
 - Explain the plan to address the root cause and restore fresh data.
 - e. Monitor the restored mart:

- Closely monitor the restored mart to ensure it refreshes successfully and within the SLA.
- Keep an eye on downstream consumers to confirm they have access to fresh data.

5. Lessons learned:

- After successfully rolling back the changes, document the incident, the rollback process, and the lessons learned.
- Update relevant knowledge bases, onboarding materials, and monitoring/alerting setups to prevent similar issues in the future.

By identifying these top risks and having a well-defined rollback plan, you can minimize the potential damage caused by a failed restoration attempt and ensure a quick recovery in case things go awry.

Engineering counterpart**2:38 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.