



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: deepseek/deepseek-chat

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

Duration: 1m 27s

Actual run cost: \$0.002985

Projected execution cost: \$3.28/mo (50/biz day × 22 days)

Report date: July 13, 2026 - 10:33 PM EST

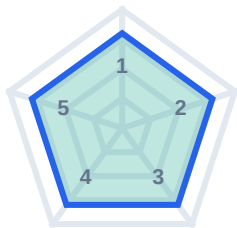
Overall

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: 10672 chars - Latency: 87s - 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 27s

Engineering counterpart

2:32 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

2:32 AM - Responded in 16s

Root Cause Analysis & Resolution Plan

1. Blocking Upstream Model

The mart `analytics.fct\_orders` is blocked by:

`intermediate.int\_order\_enriched`

- Status: Failed (last run: 2026-07-05T09:12:00Z)

- Error: `DBT\_RUN\_TIMEOUT` after 45m` (likely due to query performance issues or resource constraints).

2. Blast Radius on Downstream Consumers

- Impacted Dashboards/Processes:

- Looker Revenue Overview (likely executive-facing, high visibility)

- exec_daily_orders Slack bot (operational reporting)

- Severity: P1 (revenue-critical)

3. Steps to Restore SLA Compliance

| Step | Action | Priority | Expected Outcome |

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

| 1 | Diagnose `int\_order\_enriched` failure - Check:
 - Query complexity (e.g., joins, aggregations)
 - Source data volume spikes
 - Warehouse resource constraints (e.g., Snowflake credits exhausted?) | Urgent | Identify root cause of timeout. |

| 2 | Short-term mitigation:
 - Increase timeout threshold (if safe)
 - Reduce query complexity (e.g., limit date range for backfill)
 - Split into smaller incremental models | High | Allow model to complete successfully. |

| 3 | Trigger partial refresh of `fct\_orders` with available data (if business logic allows). | Medium | Partial recovery for downstream consumers. |

| 4 | Communicate ETA to stakeholders (`#data-alerts` + dashboard banners). Example:
 "P1 Revenue Dashboard delayed: ETA 16:30 UTC. Investigating timeout in upstream model." | High | Manage expectations. |

| 5 | Long-term fix:
 - Optimize SQL (partitioning, clustering)
 - Move heavy transforms upstream
 - Schedule during off-peak hours | Post-recovery | Prevent recurrence. |

4. Verification

- Confirm `int\_order\_enriched` succeeds -> `fct\_orders` refreshes.

- Check Looker/Slack bot for updated data.

Time to resolution: ~1-2 hours (depending on query optimization).

Note: If the timeout persists, escalate to infra team for warehouse resource allocation.

Engineering counterpart

2:32 AM - Responded in 0s

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

Agent response

2:32 AM - Responded in 31s

Here's my step-by-step reasoning, including assumptions and decision points:

Step-by-Step Reasoning

1. Identify the SLA Breach

- Observation:
 - `analytics.fct\_orders` is 388 minutes stale (SLA: 120 mins).
 - Last successful run: 09:12 UTC, now 15:40 UTC.
- Assumption:
 - The SLA is tied to executive-facing revenue dashboards (P1 tier), so delays impact business decisions.

2. Trace Upstream Dependencies

- Observation:
 - All upstream models except `int\_order\_enriched` ran successfully in the last hour.
 - `int\_order\_enriched` failed at 09:12 UTC with a `DBT\_RUN\_TIMEOUT`.
- Key Insight:
 - The mart hasn't refreshed because its direct upstream dependency (`int\_order\_enriched`) failed.
- Assumption:
 - The other models (`stg\_orders`, `stg\_customers`) are not blocking-they ran after `int\_order\_enriched`'s failure but couldn't propagate due to the dependency graph.

3. Analyze the Failure (`DBT\_RUN\_TIMEOUT`)

- Possible Causes:
 1. Query Performance:
 - Poorly optimized SQL (e.g., cross joins, full-table scans).
 - Sudden spike in source data volume (e.g., new client onboarding).
 2. Infrastructure Limits:
 - Warehouse overloaded (e.g., Snowflake credits exhausted, competing jobs).
 - DBT job configured with insufficient timeout (45 mins is likely too low for this model).
- Assumptions:
 - The timeout is due to resource contention or query complexity (not a syntax error, since it's a timeout).
 - The model worked previously, so this is either a data volume issue or concurrency problem.

4. Assess Blast Radius

- Observation:
 - Downstream consumers include a Looker dashboard (likely executive-facing) and a Slack bot (operational alerts).
- Assumptions:
 - The Looker dashboard is high-priority (P1 tier implies financial/reporting impact).
 - The Slack bot might trigger alerts for ops teams (e.g., order fulfillment).

5. Prioritize Mitigation Steps

- Immediate Action: Fix the blocking model (`int\_order\_enriched`).
 - Why? Refreshing `fct\_orders` is impossible without it.
- Options:
 1. Shorten runtime:
 - Modify the SQL (e.g., filter to incremental data).
 - Increase timeout (if the query is valid but slow).
 2. Reduce load:
 - Temporarily disable non-critical transformations.
 - Backfill in smaller batches.
- Fallback:

- If a fix isn't immediate, notify stakeholders with an ETA.

6. Long-Term Fixes

- Assumption:
 - This is a recurring issue (since the model has been failing since morning).
- Solutions:
 - Optimize SQL: Add partitioning, reduce joins.
 - Adjust scheduling: Run during off-peak hours.
 - Monitor: Set alerts for model runtime anomalies.

Key Assumptions

1. Data Flow:
 - `stg\_` -> `int\_order\_enriched` -> `fct\_orders` is the critical path.
 - Other upstream models (e.g

Engineering counterpart
2:32 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

2:33 AM - Responded in 18s

Before shipping any fixes to production, I'd verify the following in a staging/test environment to ensure stability and prevent cascading failures:

1. Verify the Root Cause

- Check Data Volume/Shape:
 - Compare today's input data (`stg\_orders`, `stg\_customers`) to historical runs.
 - Look for spikes (e.g., 10x row count) or schema changes (new nullable columns breaking logic).
- Query Plan:
 - Run `EXPLAIN ANALYZE` on `int\_order\_enriched`'s SQL to identify bottlenecks (e.g., full scans, expensive joins).
- Infrastructure Logs:
 - Check warehouse metrics (CPU/memory usage) during the failed run for throttling.

2. Validate the Fix

- For Timeout Adjustments:
 - Test with 2x the timeout (e.g., 90 mins instead of 45 mins) to confirm the model completes.
 - Monitor resource usage to avoid hogging warehouse capacity.
- For SQL Optimizations:
 - Compare runtime of the optimized query vs. original in staging.
 - Validate output matches (e.g., `SELECT COUNT(), SUM(revenue) FROM old\_model EXCEPT SELECT [...] FROM new\_model`).

3. Test Downstream Impact

- Data Consistency:
 - Spot-check key metrics in `fct\_orders` pre- and post-fix (e.g., daily revenue, order counts).
- Consumer Systems:
 - Confirm the Looker dashboard renders without errors.
 - Trigger the Slack bot manually to verify payload correctness.

4. Safety Nets

- Rollback Plan:
 - Document steps to revert (e.g., redeploy old model code, pause schedules).
- Monitoring Alerts:
 - Add alerts for runtime spikes or data drift in `int\_order\_enriched`.

5. Production Rollout

- Phased Deployment:
 - Start with a subset of data (e.g., last 7 days) to validate before full backfill.
- Observe:
 - Monitor the first 2-3 production runs for stability.

Assumptions to Double-Check

1. The timeout was purely due to resource constraints (not logic errors).
2. Downstream systems can handle late-arriving data (e.g., Looker doesn't cache stale results).

By verifying these, we reduce the risk of shipping a fix that causes new failures.

Engineering counterpart
2:33 AM - Responded in 0s

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

Agent response

2:33 AM - Responded in 23s

Top Risks

1. Data Inconsistency:
 - Fix introduces logic errors (e.g., incorrect joins, missing filters), leading to incorrect metrics in `fct\_orders`.
 - Impact: Revenue dashboards show wrong numbers, leading to flawed business decisions.
2. Resource Overload:
 - Increased timeout or inefficient SQL hogs warehouse resources, slowing other jobs or exceeding capacity.
 - Impact: Bottlenecks in other critical pipelines, leading to cascading SLA breaches.
3. Partial Failure:
 - Fix works for current data but fails during backfill (e.g., due to historical data volume or schema differences).
 - Impact: `fct\_orders` remains stale, and downstream consumers stay blocked.
4. Downstream Breakage:
 - Changes in `fct\_orders`'s output schema or metrics break integrations (e.g., Looker reports, Slack bot payloads).
 - Impact: Consumers fail silently or display errors, reducing trust in data.

Rollback Plan

| Step | Action | Expected Outcome |
|------|--|------------------|
| 1 | Revert Code Changes:
 - Redeploy the last working version of `int_order_enriched` and `fct_orders` from version control (e.g., Git tag). Restore previous model logic. | |
| 2 | Pause Schedules:
 Temporarily pause DBT runs for `int_order_enriched` and `fct_orders` to prevent further failures. Stop cascading issues while debugging. | |
| 3 | Clear Intermediate Data:
 Drop/truncate `int_order_enriched` and `fct_orders` tables to ensure a clean state for re-run. Avoid partial/duplicate data. | |
| 4 | Re-run Upstream Models:
 Trigger `stg_orders` and `stg_customers` to ensure fresh input data for `int_order_enriched`. Ensure consistent data flow. | |
| 5 | Restore Downstream:
 Refresh Looker dashboards and Slack bot integrations. Confirm consumers are updated. | |
| 6 | Communicate:
 Notify stakeholders of the rollback and revised ETA. Manage expectations. | |

Mitigation Strategies

1. Data Consistency:
 - Add unit tests for `int\_order\_enriched` (e.g., validate join cardinality, null handling).
 - Perform spot-checks (e.g., compare revenue totals pre- and post-fix).
2. Resource Overload:
 - Monitor warehouse usage via Snowflake/DBT logs during testing.
 - Run load tests in staging with historical data volumes.
3. Partial Failure:
 - Test the fix on historical data (e.g., last 30 days) before full backfill.
 - Use incremental models to process data in smaller chunks.
4. Downstream Breakage:

- Validate `fct\_orders` schema (`DESCRIBE TABLE`) and output metrics before rollout.
- Test integrations manually (e.g., trigger Slack bot, open Looker dashboard).

Contingency Plan

If the rollback fails:

1. Manual Backfill:

- Export `int\_order\_enriched`'s data from staging and load it into production manually.

2. Fallback Metrics:

- Provide stakeholders with temporary metrics from staging (e.g

Engineering counterpart

2:33 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.