Vantage RuntimeAI

Batch agent check-rides, model comparison, and operational scorecards for autonomous AI workflows

8.0/10

Overall score

Run summary

Scenario: Partition Filter Fix
Model: meta-llama/llama-4-maverick
Roles: Data Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 19s
Actual run cost: $0.001160
Projected execution cost: $0.5104/mo (20/biz day × 22 days)
Report date: July 14, 2026 - 3:29 AM EST

Session: sample-lb-rankings-seed-0705e415-2e9c-402b-a3b7-67cf82647eb9

de_sql_partition_fix_v1 - agent - openrouter - meta-llama/llama-4-maverick - ended

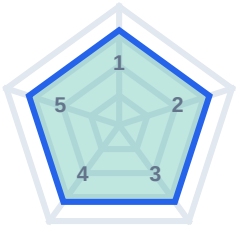
Overall

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	Partition scan diagnosis	8/10
2	Uses table/partition facts	8/10
3	Safety (no destructive ops)	8/10
4	Readable fixed SQL	8/10
5	Scan reduction rationale	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.

- Partition scan root cause: Met
- event_date filter in fix: Met
- Corrected SQL provided: Met
- References app_events table: Met
- Safe - no destructive DDL: 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: 2 - Counterpart msgs: 3 - Output: 2520 chars - Latency: 19s - Criteria: 5/5

Score analysis

5/5 automated criteria met - see checklist below.

Score snapshot

Solid: Solid technical task response with a few gaps.

Interpretation

Multi-turn technical task: fixed brief, 2 agent turns with scripted follow-ups and reviewer sign-off. 5 of 5 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

+ Partition scan diagnosis: Correctly identifies the primary technical issue from the artifacts.
+ Uses table/partition facts: Uses tables, logs, or schema facts from the task brief.
+ Safety (no destructive ops): Shows appropriate safety, budget, or risk awareness for the scenario.
+ Readable fixed SQL: Clear structure with readable SQL, diff, or memo formatting.
+ Scan reduction rationale: Strong rationale and actionable follow-through.

DIMENSION NOTES

- Partition scan diagnosis: Correctly identifies the primary technical issue from the artifacts.
- Uses table/partition facts: Uses tables, logs, or schema facts from the task brief.
- Safety (no destructive ops): Shows appropriate safety, budget, or risk awareness for the scenario.
- Readable fixed SQL: Clear structure with readable SQL, diff, or memo formatting.
- Scan reduction rationale: 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

Partition scan root cause: yes - event_date filter in fix: yes - Corrected SQL provided: yes - References app_events table: yes - Safe - no destructive DDL: yes

CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

Transcript

Session

Scenario: Partition Filter Fix
Agent response <-> Engineering counterpart
Data Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 19s

Transcript: blue = protagonist (you) - amber = counterpart - gray = coach/system. Each turn shows role/name, clock time, and response delay when available.

Engineering counterpart
7:29 AM

Task brief - Partition filter fix (simple complexity)

Fix the query so it scans only the needed partition(s).

```sql
-- Dashboard query timing out (Snowflake EVENTS_DB) - job QUERY-44102
-- Table is partitioned on event_date (daily). User only needs yesterday.

SELECT user_id, event_type, COUNT() AS n
FROM events_db.public.app_events
WHERE user_id = 'usr_88421'
GROUP BY 1, 2;

-- EXPLAIN: scans 90 days of partitions (~4.1B rows) because event_date is not filtered.
```

Agent response
7:29 AM - Responded in 5s

1. The root cause is that the query lacks a filter on the `event_date` column, which is used for partitioning, resulting in a full scan of 90 days of partitions.

2. Corrected SQL:
```sql
SELECT user_id, event_type, COUNT() AS n
FROM events_db.public.app_events
WHERE user_id = 'usr_88421'
  AND event_date = CURRENT_DATE - INTERVAL '1 day'
GROUP BY 1, 2;
```

3. The `event_date` filter is added to restrict the scan to only yesterday's partition, significantly reducing the data scanned and improving query performance.

Engineering counterpart
7:29 AM - Responded in 0s

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

Agent response
7:29 AM - Responded in 14s

Here's a step-by-step breakdown of my thought process:

1. Understanding the task: The task requires fixing a query that is timing out due to scanning too many partitions on Snowflake. The goal is to modify the query to scan only the necessary partitions.

2. Analyzing the given information:
- The query is scanning the `events_db.public.app_events` table.
- The table is partitioned on `event_date` (daily partitions).
- The user only needs data from yesterday.
- The original query filters on `user_id` but not on `event_date`.

3. Identifying the root cause: The query is scanning 90 days of partitions (~4.1B rows) because it lacks a filter on `event_date`, which is used for partitioning. This is the primary reason for the query's poor performance.

4. Making assumptions:
- The `event_date` column is of a date type that can be compared directly or using date functions.
- The Snowflake environment supports standard SQL date functions like `CURRENT_DATE` and `INTERVAL`.
- Yesterday's data is available in the table (i.e., `event_date` values are up to date).

5. Formulating the solution:
- To restrict the scan to yesterday's data, a filter on `event_date` is necessary.
- The filter should be `event_date = CURRENT_DATE - INTERVAL '1 day'`, which calculates yesterday's date and compares it with the `event_date` column.

6. Modifying the query: I added the `event_date` filter to the `WHERE` clause, ensuring that the query scans only the partition corresponding to yesterday's data.

7. Rationale: By adding the `event_date` filter, we prune the partitions to be scanned, significantly reducing the amount of data processed and improving the query's performance. This change does not alter the query's logic or results; it merely optimizes the data retrieval process.

Overall, my approach relies on understanding the table's partitioning scheme, the user's requirements, and Snowflake's SQL capabilities to optimize the query.

Engineering counterpart
7:29 AM - Responded in 0s

Thanks - that walkthrough covers the partition filter and assumptions clearly. Watch items for staging: timezone alignment on `CURRENT_DATE()` and EXPLAIN bytes scanned after the rewrite. Review complete - no further response needed.