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: deepseek/deepseek-v4-flash
Roles: Data Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 20s
Actual run cost: $0.000616
Projected execution cost: $0.2710/mo (20/biz day × 22 days)
Report date: July 14, 2026 - 3:28 AM EST

Session: sample-lb-rankings-seed-7657d7dd-36e3-4a58-94c7-1f37fb1e1f7e

de_sql_partition_fix_v1 - agent - openrouter - deepseek/deepseek-v4-flash - ended

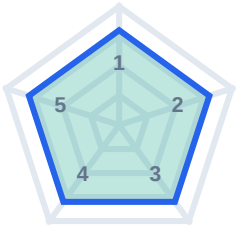
Overall

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: 3014 chars - Latency: 20s - 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: 20s

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

Engineering counterpart
7:27 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
7:28 AM - Responded in 10s

- Root cause: Missing `event_date` filter caused Snowflake to scan all partitions (90 days) instead of only the single day needed.

- 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 - 1
GROUP BY 1, 2;
```

- Rationale: Adding `event_date = CURRENT_DATE - 1` restricts the scan to yesterday's partition, eliminating the full table scan and reducing the row count from ~4.1B to a single day's data.

Engineering counterpart
7:28 AM - Responded in 0s

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

Agent response
7:28 AM - Responded in 10s

Let's break this down step by step.

Step 1 - Understand the symptom
The query times out, scanning around 4.1 billion rows across 90 daily partitions. This immediately tells us that partition pruning is not happening - the engine has to read every partition.

Step 2 - Examine the query
```sql
SELECT user_id, event_type, COUNT() AS n
FROM events_db.public.app_events
WHERE user_id = 'usr_88421'
GROUP BY 1, 2;
```
There is no filter on the partition key (`event_date`). Even though `user_id` is selective, Snowflake cannot skip partitions without a filter on the partition column.
Snowflake uses static partition pruning - it evaluates filters at compile time. A filter on a non?partition column cannot prune partitions.

Step 3 - Identify the needed partition
The user's requirement is "only yesterday" (as stated in the brief).
That means we want exactly one partition: `event_date = (today - 1 day)`.

Step 4 - Assumptions I made
1. Partition column is `event_date` - given in the brief: "Table is partitioned on `event_date` (daily)."
2. `event_date` is a `DATE` type (or at least comparable to `CURRENT_DATE - 1`). This is standard for Snowflake daily partitions.
3. Yesterday's data exists - otherwise the query would return zero rows, but it would still prune correctly.
4. `CURRENT_DATE` returns the date in the session time zone - the filter should match the partition boundary. If the table uses a different time zone, an explicit cast or `DATEADD` might be safer, but for simplicity we assume local time is fine.
5. No other filters are needed - the `GROUP BY` and `COUNT()` are correct for the dashboard requirement.
6. The partition scheme is `BY DATE` - common practice for daily data.

Step 5 - Fix the query
Add the filter:
```sql
AND event_date = CURRENT_DATE - 1
```
This tells Snowflake: "only read the partition where `event_date` equals yesterday's date."
Snowflake's optimizer then prunes the other 89 partitions, reducing the scan from 4.1B rows to ~45 million (or whatever a single day holds).

Step 6 - Verify the fix
With the new filter, the `EXPLAIN` plan should show only one partition scanned, and the execution time drops from time?out to a few seconds.

Final takeaway:
The root cause was missing the partition column filter. The fix adds exactly that filter, using the business requirement ("yesterday") to target a single partition.

Engineering counterpart
7:28 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.