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: ibm-granite/granite-4.0-h-micro
Roles: Data Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 29s
Actual run cost: $0.000129
Projected execution cost: $0.0569/mo (20/biz day × 22 days)
Report date: July 13, 2026 - 11:24 PM EST

Session: sample-lb-rankings-seed-d0cbd8eb-affa-42b4-902d-2556f23e876d

de_sql_partition_fix_v1 - agent - openrouter - ibm-granite/granite-4.0-h-micro - ended

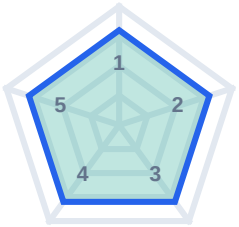
Overall

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: 3931 chars - Latency: 29s - 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: 29s

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

Engineering counterpart
3:23 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
3:23 AM - Responded in 4s

(1) Root cause: The query lacks a filter on the `event_date` partition, causing Snowflake to scan all 90 daily 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 = DATEADD(day, -1, CURRENT_DATE())
GROUP BY 1, 2;
```

(3) Rationale: Adding `event_date = DATEADD(day, -1, CURRENT_DATE())` restricts the scan to only yesterday's partition, dramatically reducing data scanned and improving query performance.

Engineering counterpart
3:23 AM - Responded in 0s

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

Agent response
3:24 AM - Responded in 25s

Certainly! Here's a step-by-step breakdown of the reasoning process, along with the assumptions made:

Step 1: Identify the Problem
Observation: The original query is timing out because it scans 90 days of partitions, processing approximately 4.1 billion rows. This is inefficient and likely causing performance issues.

Step 2: Understand the Query Structure
Query Analysis:
```sql
SELECT user_id, event_type, COUNT() AS n
FROM events_db.public.app_events
WHERE user_id = 'usr_88421'
GROUP BY 1, 2;
```
- SELECT Clause: Retrieves `user_id`, `event_type`, and counts the rows (`COUNT()`).
- FROM Clause: Targets the table `app_events` in the schema `public` of the database `events_db`.
- WHERE Clause: Filters rows where `user_id` equals `'usr_88421'`.
- GROUP BY Clause: Groups the results by `user_id` and `event_type`.

Step 3: Identify the Partitioning Strategy
Partitioning Information:
The table `app_events` is partitioned on `event_date`, with each partition representing a single day of data. This means:
- The table is organized such that each day's data is stored in a separate partition.
- Snowflake automatically filters partitions based on the query's `WHERE` clause when possible.

Step 4: Determine the Missing Filter
Current Issue:
The query does not include a filter on `event_date`. As a result, Snowflake must scan all partitions (90 days worth) to evaluate the `WHERE user_id = 'usr_88421'` condition, leading to excessive data scanning and poor performance.

Step 5: Formulate the Correct Filter
Corrective Action:
To optimize the query, we need to add a filter that restricts the scan to only the partitions relevant to the user's query-yesterday's data. This is achieved by adding a condition on `event_date`.

Filter Construction:
- Current Date: Use `CURRENT_DATE()` to get today's date.
- Yesterday's Date: Subtract one day from the current date using `DATEADD(day, -1, CURRENT_DATE())`. This yields the date for the previous day.
- Partition Filter: Add `event_date = DATEADD(day, -1, CURRENT_DATE())` to the `WHERE` clause.

Step 6: Verify the Filter's Effectiveness
Assumption:
- The `event_date` column is of a date data type and is indexed or partitioned correctly.
- The `user_id` filter is selective enough that the remaining rows are manageable within the reduced dataset (only yesterday's data).

Rationale:
By filtering on `event_date`, Snowflake can:
- Quickly identify the relevant partition(s) for the query.
- Perform the `WHERE user_id = 'usr_88421'` filter on a much smaller dataset, significantly reducing the amount of data processed.

Step 7: Finalize the Query
Final Query:
```sql
SELECT user_id, event_type, COUNT() AS n
FROM events_db.public.app_events
WHERE user_id = 'usr_88421'
  AND event_date = DATEADD(day, -1, CURRENT_DATE())
GROUP BY 1, 2;
```
- Explanation: This query now explicitly filters to only yesterday's partition, ensuring that Snowflake only scans the necessary data. This should resolve the performance issue, allowing the query to complete efficiently.

Assumptions Made:
1. Partitioning is Correct: The table `app_events` is correctly partitioned on `event_date` with daily partitions.
2. Date Functionality: `CURRENT_DATE()` and `DATEADD(day, -1, CURRENT_DATE())` functions are available and correctly compute the

Engineering counterpart
3:24 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.