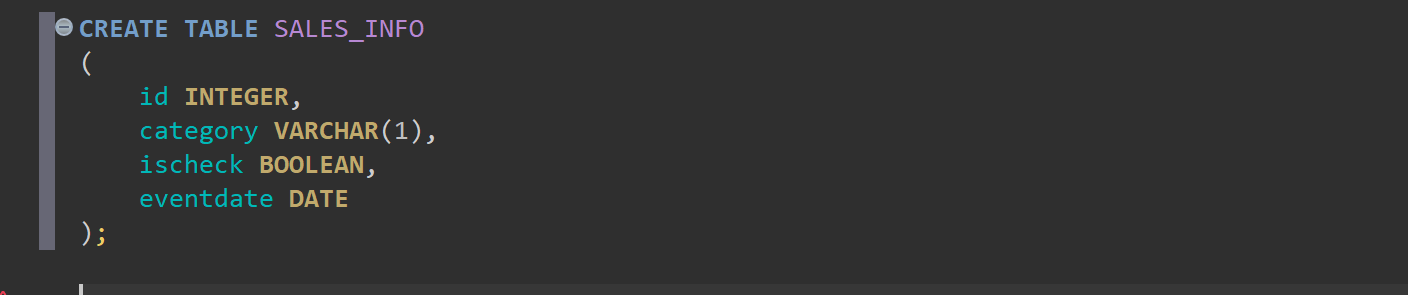
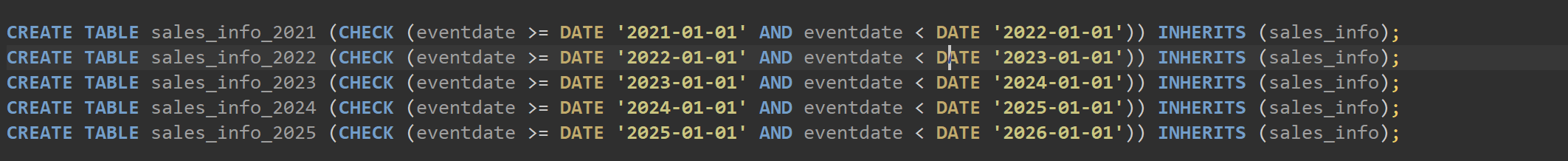
* 1. TASK 1: USE INHERITANCE

Create table:

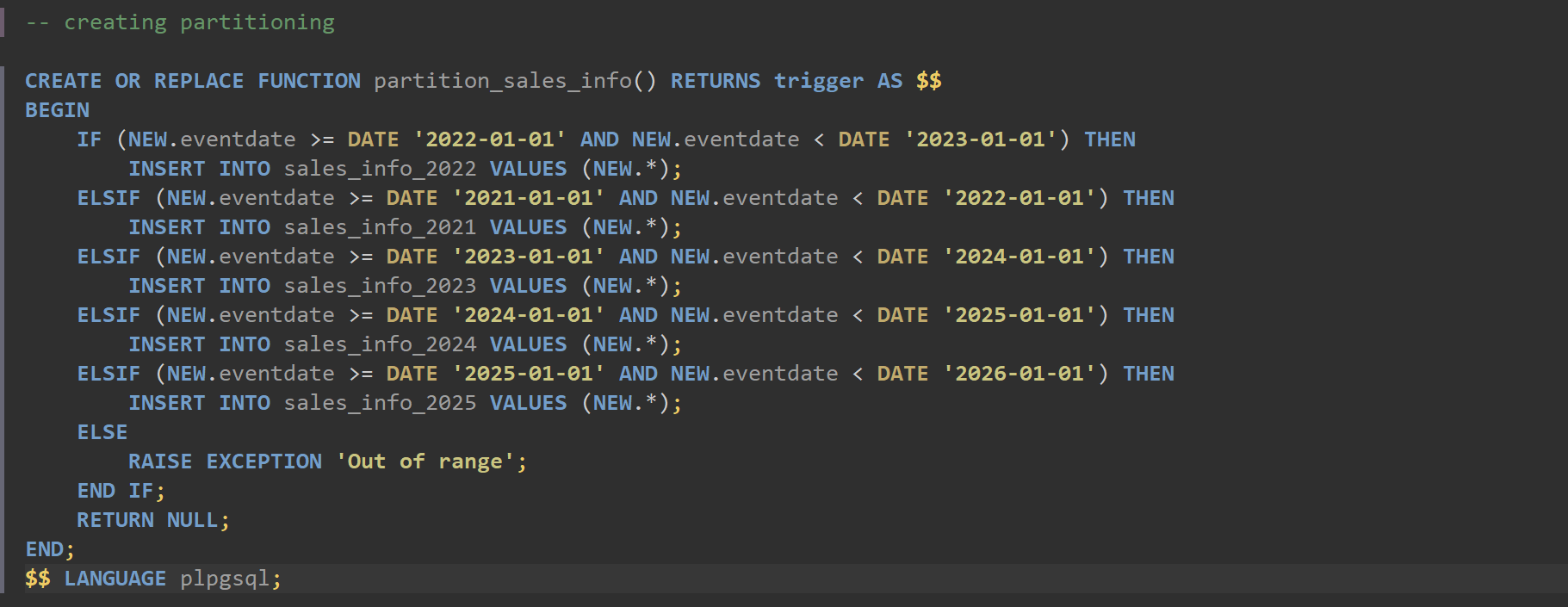


Apply partitioning by using inheritance:

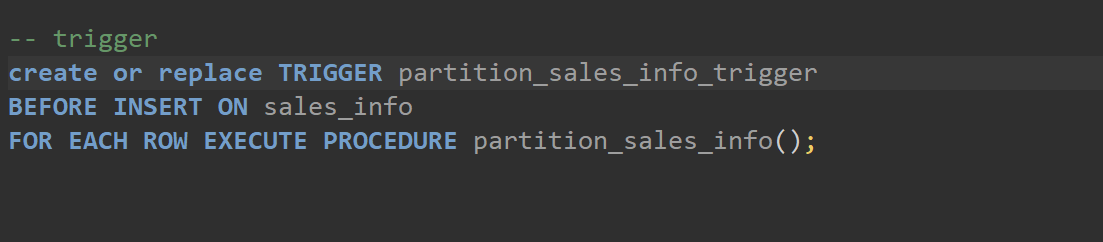
1. Create 4-5 child tables with partitioning by eventdate column. One partition is one year.



2. Create partition function for your tables. Use following as a template:



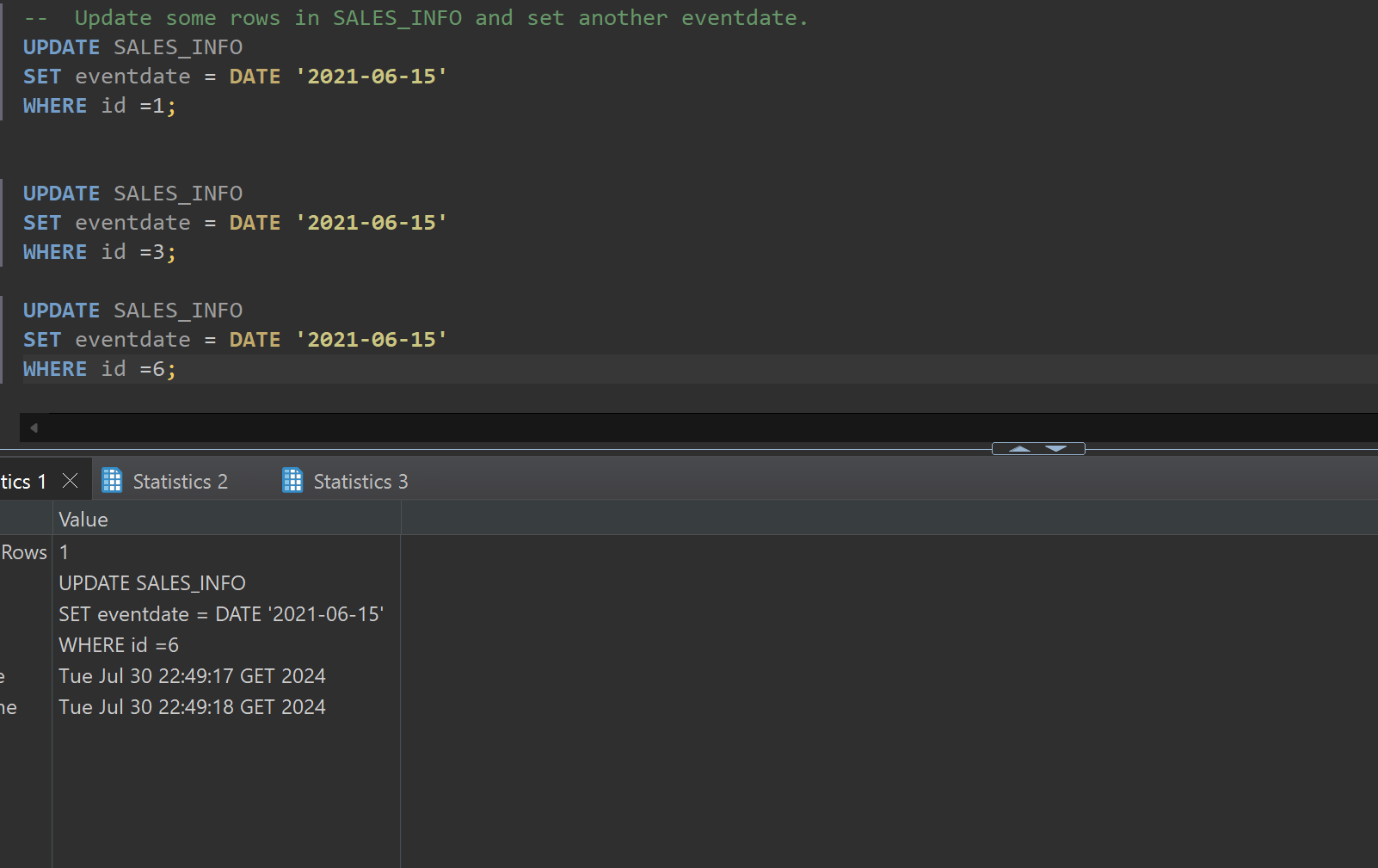
3. Create trigger for your function and tables. Use following as a template:



4. Generate test data and insert in SALES\_INFO table:



5. Update some rows in SALES\_INFO and set another eventdate.



6. Create table SALES\_INFO\_SIMPLE with the same structure as SALES\_INFO but without partitioning. Insert test data from the 5th step. Compare plans of different queries:

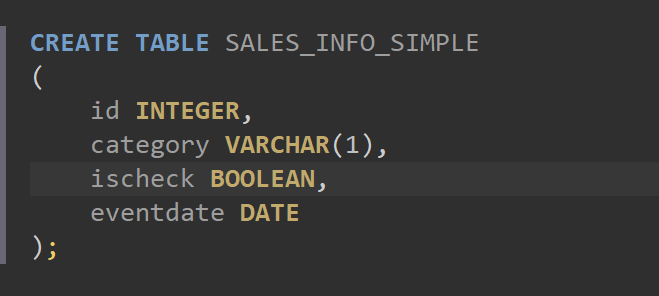
• Select all

• Select with range of dates

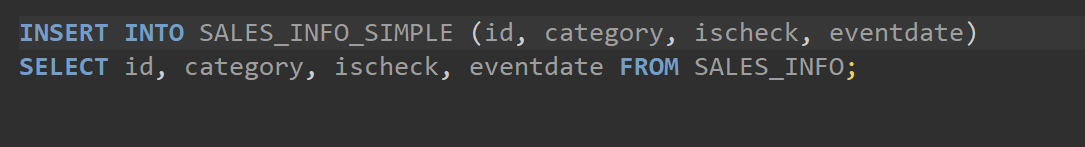
• Select exact date

• Count of all rows

• Count of rows with range of dates



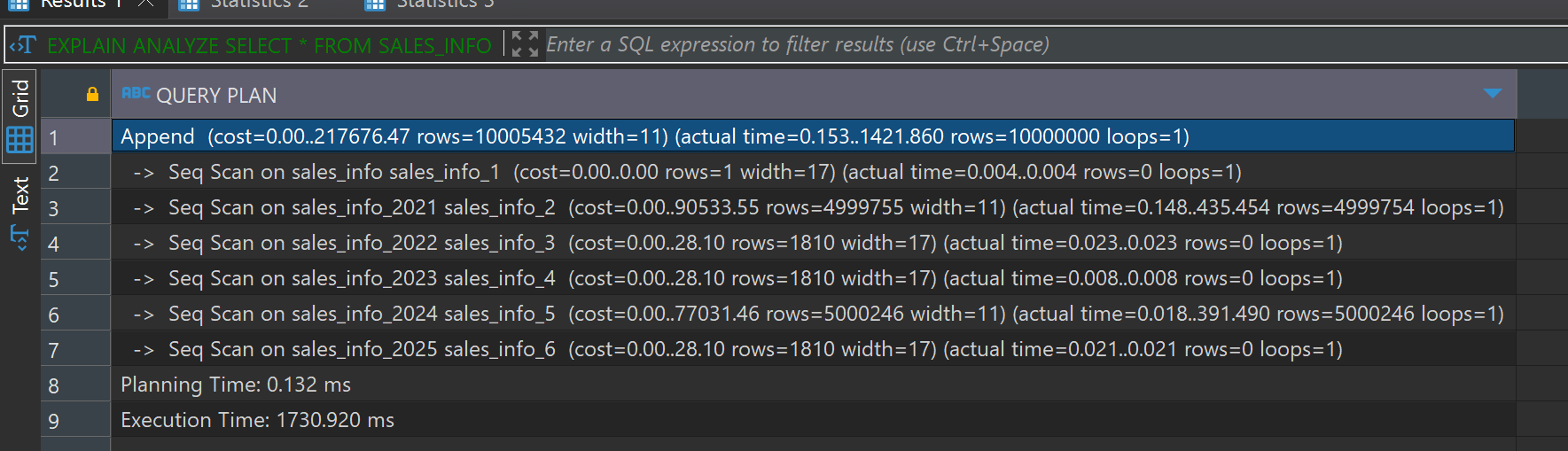
Inserting from first table into this one

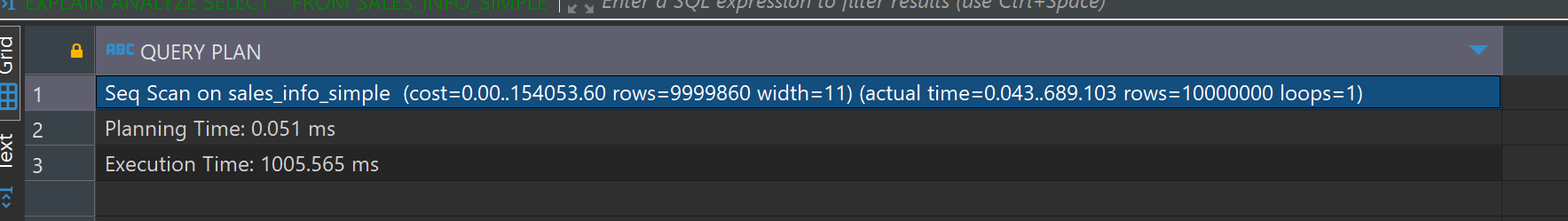


Comparing:

**EXPLAIN** **ANALYZE** **SELECT** \* **FROM** SALES\_INFO;

**EXPLAIN** **ANALYZE** **SELECT** \* **FROM** SALES\_INFO\_SIMPLE;





First of all, the non-partitioned table has a lower execution time) compared to the partitioned table for the SELECT \* query. SELECT \* query on the partitioned table has to scan multiple partitions and then combine the results, whereas the non-partitioned table performs a single scan.

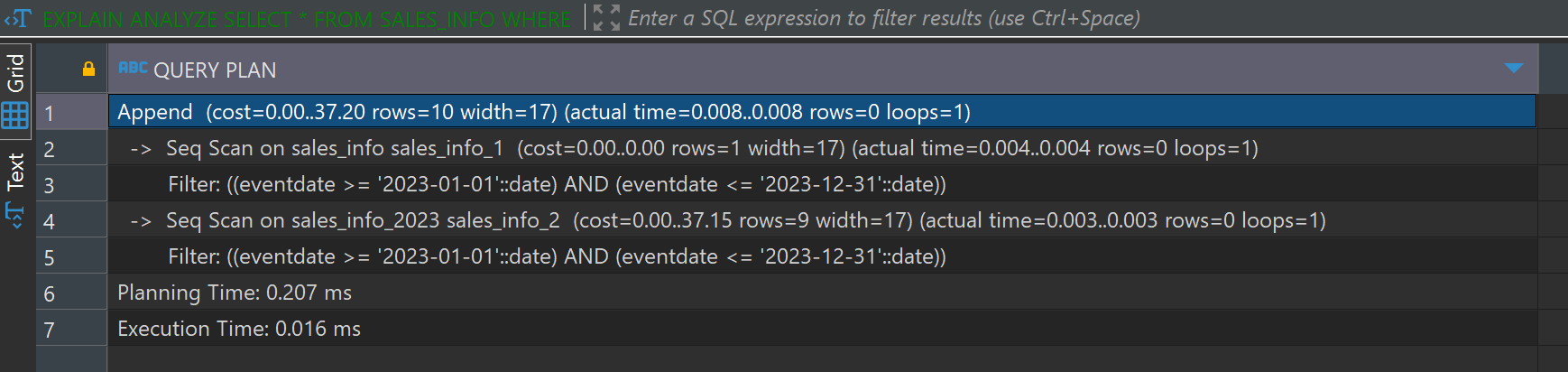
The planning time for the partitioned table is little bit higher than that of the non-partitioned table

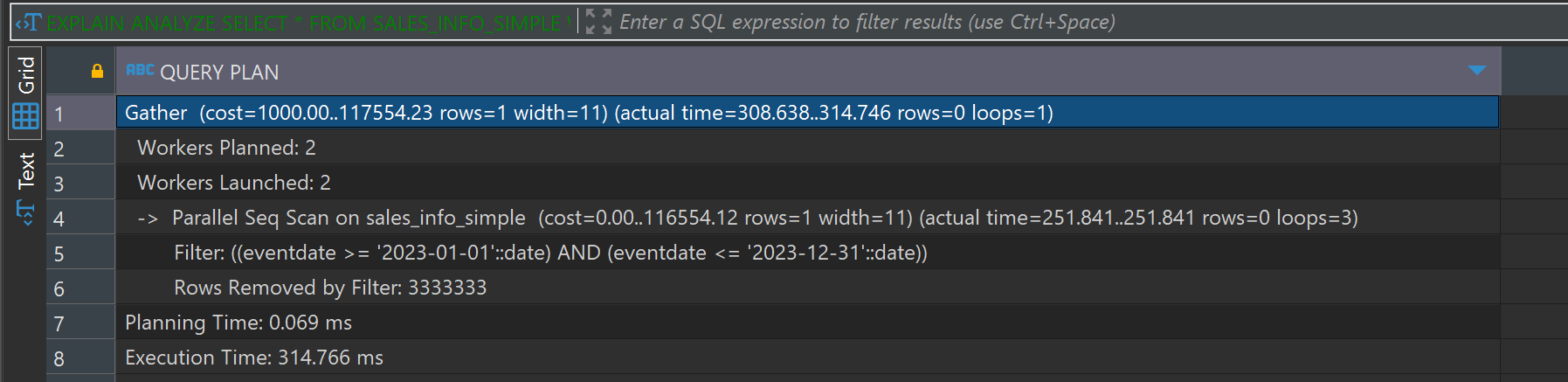
The non-partitioned table performs better in terms of execution time for full table scans, as it involves a single sequential scan.

-- select with range of dates

**EXPLAIN** **ANALYZE** **SELECT** \* **FROM** SALES\_INFO **WHERE** eventdate **BETWEEN** **'2023-01-01'** **AND** **'2023-12-31'**;

**EXPLAIN** **ANALYZE** **SELECT** \* **FROM** SALES\_INFO\_SIMPLE **WHERE** eventdate **BETWEEN** **'2023-01-01'** **AND** **'2023-12-31'**;



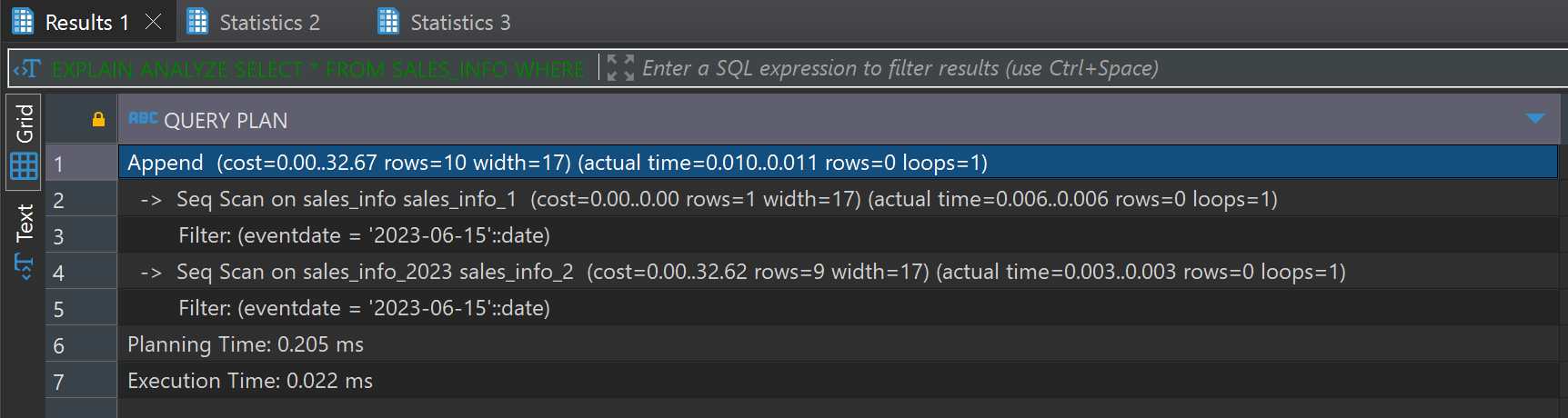


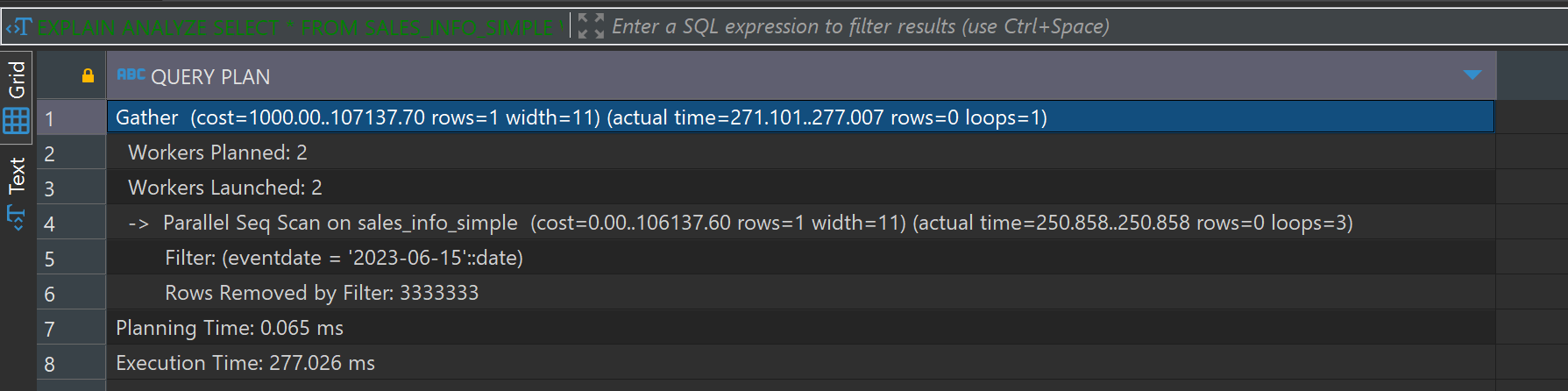
Partitioning provides significant performance benefits for range queries by limiting the scan to relevant partitions. This results in faster execution times and more efficient data retrieval. The non-partitioned table has to scan the entire table which results in higher execution times. That’s why partitioned table has a significantly lower execution time compared to the non-partitioned table for the range query.

-- Select exact date

**EXPLAIN** **ANALYZE** **SELECT** \* **FROM** SALES\_INFO **WHERE** eventdate = **'2023-06-15'**;

**EXPLAIN** **ANALYZE** **SELECT** \* **FROM** SALES\_INFO\_SIMPLE **WHERE** eventdate = **'2023-06-15'**;

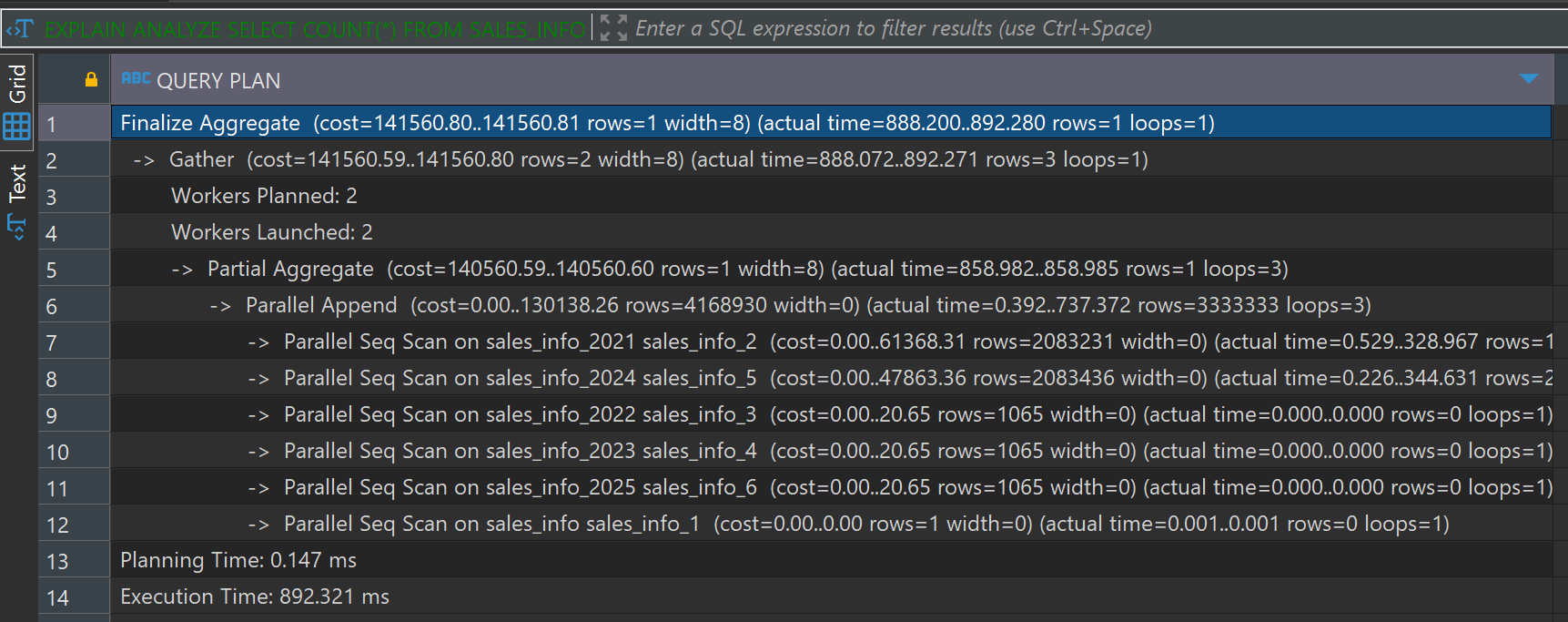


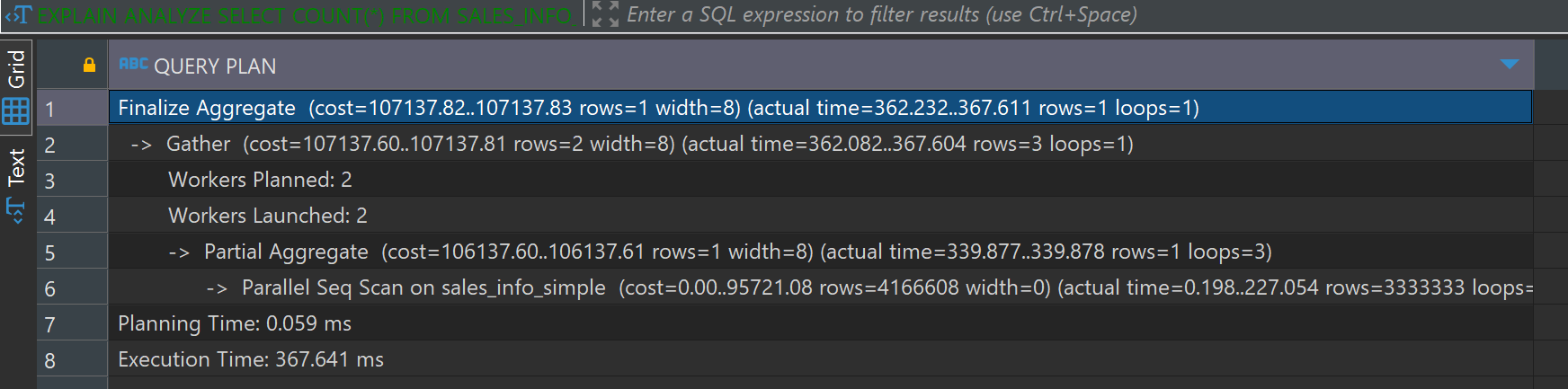


The partitioned table runs faster for the exact date query because it only looks at the needed partitions, reducing the data to scan. However, it takes a bit more time to plan the query because it has to figure out which partitions to use. The partitioned table is more efficient as it scans just the relevant partition (sales\_info\_2023). On the other hand, the non-partitioned table scans the entire table, which takes more time since it has to go through more data.

**EXPLAIN** **ANALYZE** **SELECT** **COUNT**(\*) **FROM** SALES\_INFO;

**EXPLAIN** **ANALYZE** **SELECT** **COUNT**(\*) **FROM** SALES\_INFO\_SIMPLE;



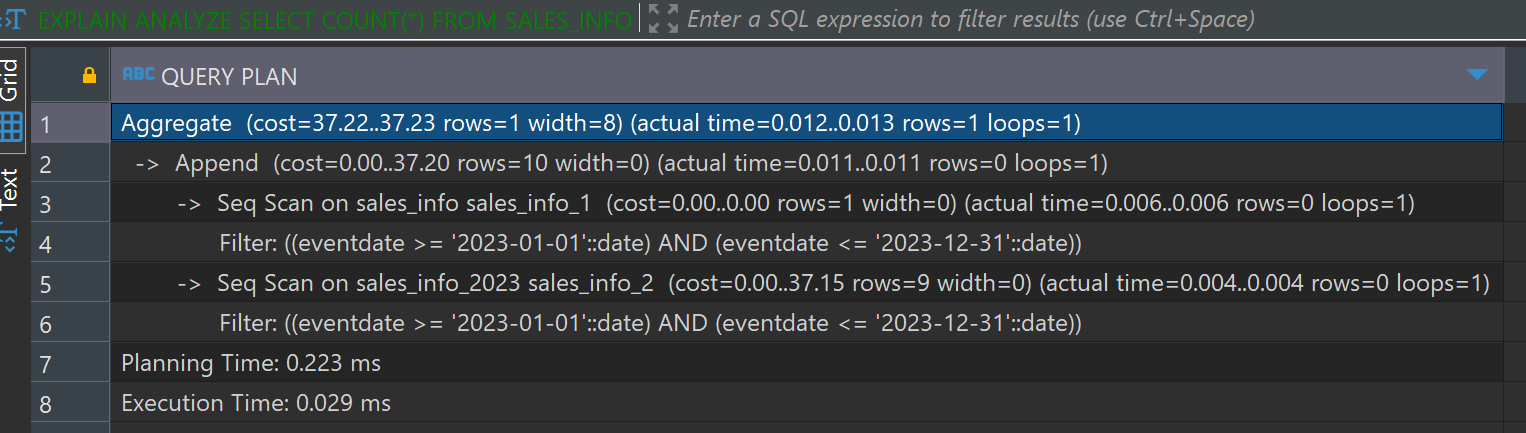


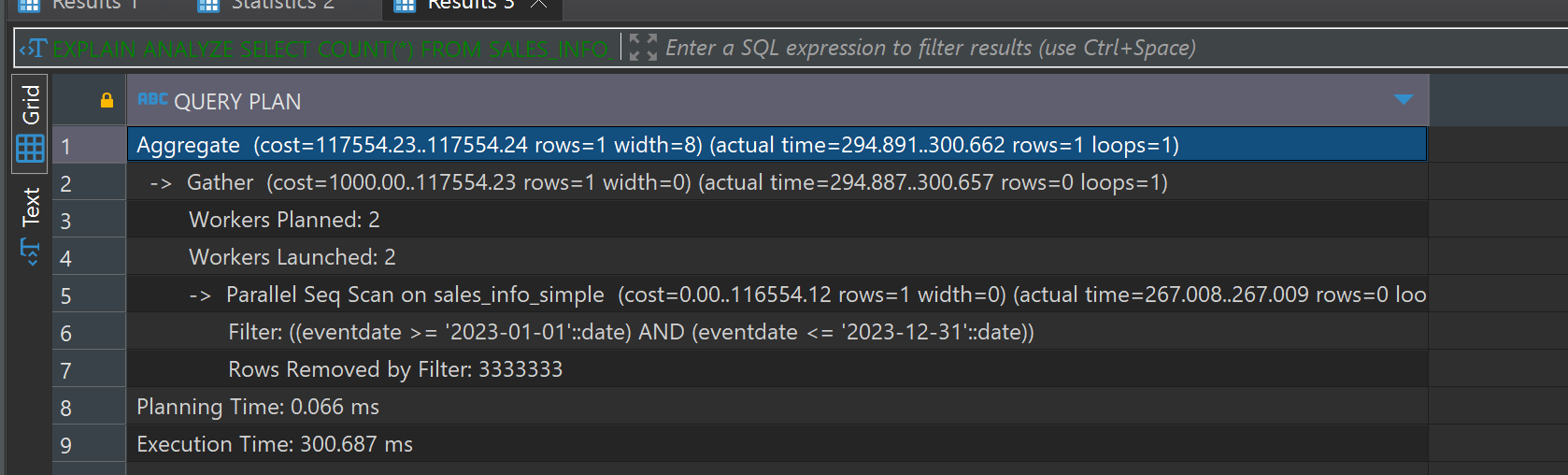
The non-partitioned table has a lower execution compared to the partitioned table. This is because the partitioned table needs to combine results from multiple partitions, which adds overhead because of it partitioned table has a higher planning time compared to the non-partitioned table. This is due to the query planner having to identify and prepare for scanning multiple partitions. the partitioned table involves multiple Parallel Seq Scan operations on each partition and that increases the total execution time. The non-partitioned table, although scanning the entire table, benefits from a more straightforward and unified scan process.

-- Count of rows with range of dates

**EXPLAIN** **ANALYZE** **SELECT** **COUNT**(\*) **FROM** SALES\_INFO **WHERE** eventdate **BETWEEN** **'2023-01-01'** **AND** **'2023-12-31'**;

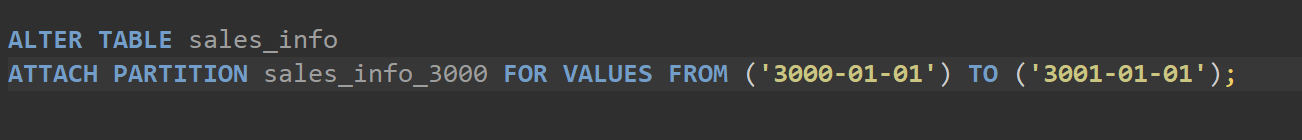
**EXPLAIN** **ANALYZE** **SELECT** **COUNT**(\*) **FROM** SALES\_INFO\_SIMPLE **WHERE** eventdate **BETWEEN** **'2023-01-01'** **AND** **'2023-12-31'**;





The SALES\_INFO has a significantly lower execution time for counting rows within a date range, as it only scans the relevant partitions. The non-partitioned table has a much higher execution time since it performs a full table scan and applies the date filter to all rows. The partitioned approach is more efficient for range queries due to targeted scans of relevant data segments.

I dropped the table sales\_info\_2021 and then created new table sales\_info\_3000



But I throws error;

Table is not partitioned,