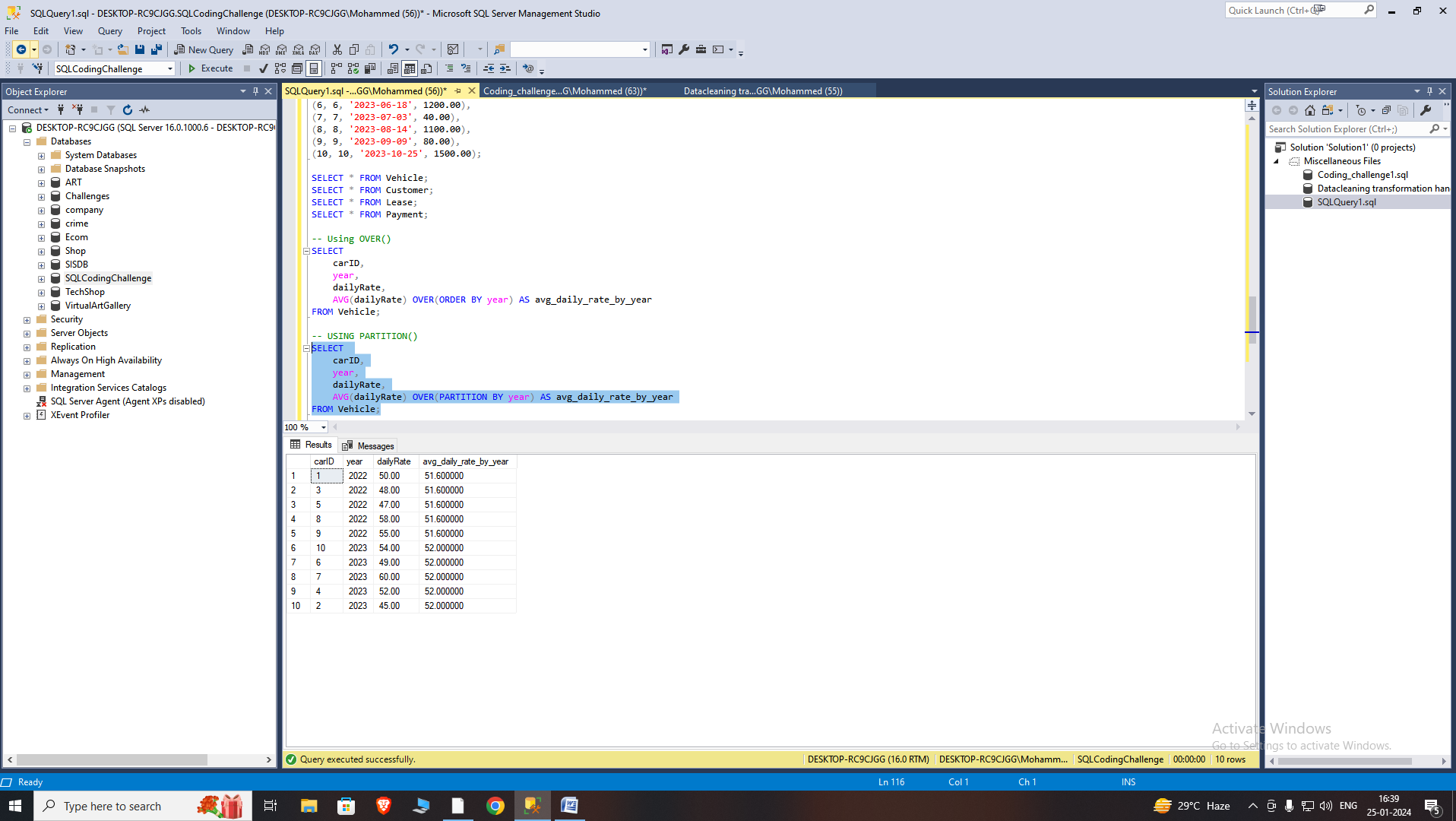
**Total Aggregations using SQL Queries, OVER and PARTITION BY Clause in SQL Queries:**

In SQL, the OVER and PARTITION BY clauses are used in conjunction with aggregate functions to perform total aggregations on specific partitions of the result set. In addition to SUM(), you can use other aggregate functions like AVG(), COUNT(), MIN(), or MAX() with OVER and PARTITION BY.



**Snowflaking:**

Snowflaking is a database normalization technique that involves breaking down dimension tables into more normalized structures by decomposing hierarchies or attributes into separate related tables. The name "snowflaking" comes from the visual representation of the schema, which resembles a snowflake with its many branches.

Characteristics:

Normalization: Snowflaking leads to a more normalized database schema, reducing redundancy and improving data integrity.

Complexity: While it reduces redundancy, it introduces complexity to queries due to the need for joining multiple tables.

Hierarchical Structure: Commonly applied to hierarchical dimensions, such as geographical hierarchies (Country > State > City) or organizational hierarchies.

**Star Schema:**

A Star Schema is a database design where a central fact table is connected to multiple dimension tables. The fact table contains quantitative data (measurements or metrics), while the dimension tables contain descriptive information related to the fact data. The schema is named for its visual resemblance to a star, with the fact table at the center and dimensions radiating outward.

Characteristics:

Denormalization: Star schemas are typically denormalized, meaning that redundant data may exist in dimension tables to improve query performance.

Simplicity: Queries are simplified due to fewer joins, making it easier to understand and faster to execute.

Fact Table: Contains primary keys from dimension tables and measures (quantitative data).

Comparison:

**Snowflaking vs. Star Schema:**

Snowflaking : Normalized, reduced redundancy, complex queries.

Star Schema: Denormalized, potential redundancy, simplified queries.

Use Cases:

Snowflaking: Suitable for scenarios where data integrity and normalization are critical, and query complexity is acceptable.

Star Schema: Commonly used in data warehousing and business intelligence applications where query performance is a priority.

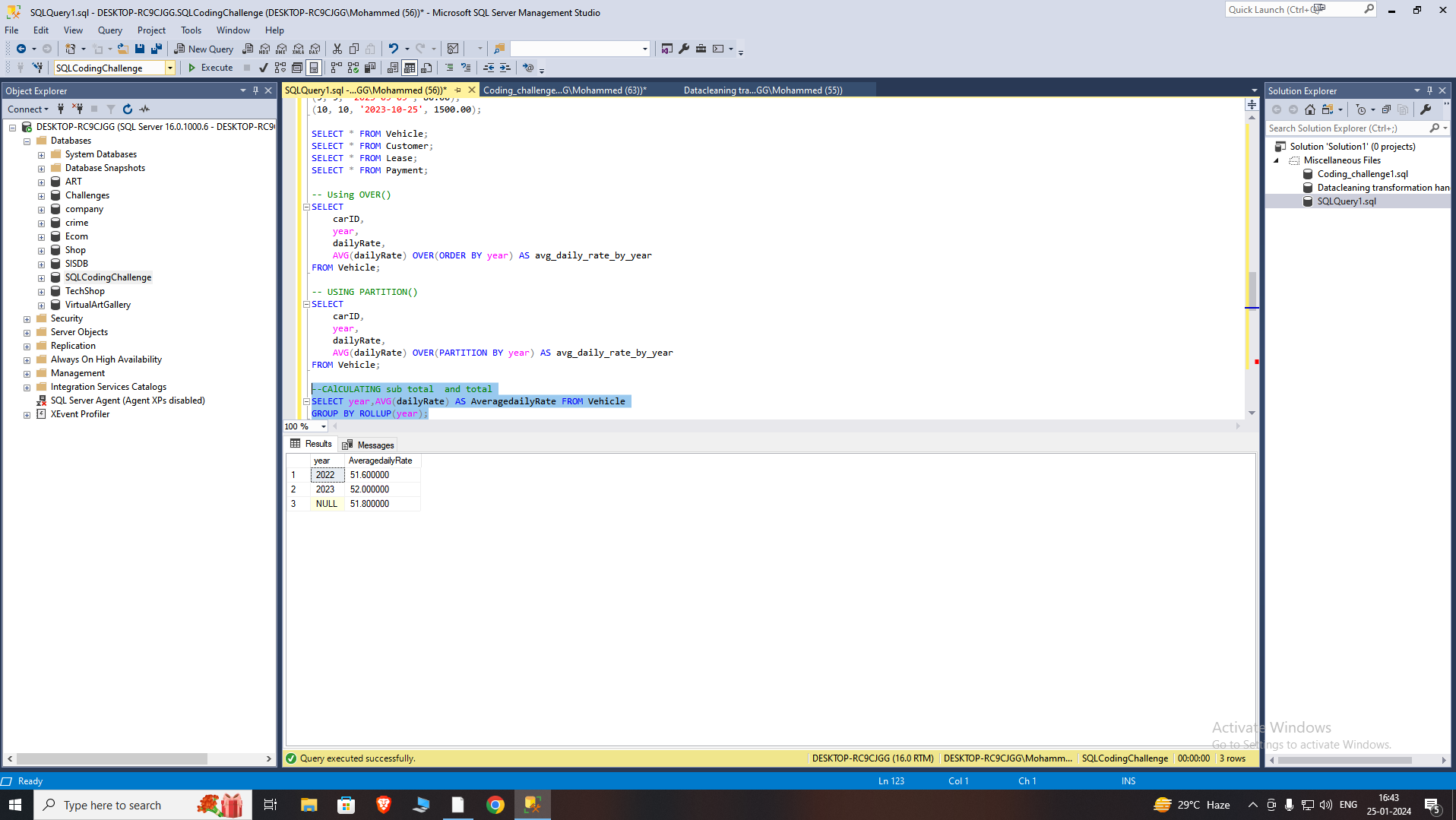
**Order of Execution of SQL Queries**

The order of execution of SQL queries is crucial to understanding how the database processes and retrieves data. The typical order of execution is as follows:

1. **FROM clause** : Specifies the tables or views from which the data will be retrieved. Defines the source data.
2. **WHERE clause** :Filters the rows based on specified conditions. Reduces the dataset to only those rows that meet the specified criteria.
3. **GROUP BY clause**: Groups rows that have the same values in specified columns into summary rows. Used with aggregate functions like COUNT, SUM, AVG, etc.
4. **HAVING clause**: Filters groups based on specified conditions. Similar to the WHERE clause but used with aggregated data after the GROUP BY clause.
5. **SELECT clause**: Specifies the columns to be retrieved. Calculates expressions or performs operations on the data.
6. **ORDER BY clause**: Sorts the result set based on specified columns and sort order.Applied after the SELECT, GROUP BY, and HAVING clauses.

**Subtotals in SQL Queries:**

In SQL, you can use the ROLLUP operator to generate subtotals in your result set. The ROLLUP operator is typically used in conjunction with the GROUP BY clause to produce subtotal and total rows for specified columns.



**Differences Between UNION EXCEPT and INTERSECT Operators in SQL Server :**

The UNION, EXCEPT, and INTERSECT operators are used in SQL to combine or compare the results of two or more SELECT statements. Here are the key differences between these operators in SQL Server:

**UNION Operator:**

Purpose: Combines the result sets of two or more SELECT statements into a single result set.

Duplicates: Removes duplicate rows from the result set.

Syntax:

SELECT column1, column2 FROM table1

UNION

SELECT column1, column2 FROM table2;

**EXCEPT Operator :**

Purpose: Returns distinct rows from the result set of the first SELECT statement that are not present in the result set of the second SELECT statement.

Duplicates: Removes duplicate rows from the result set.

Syntax:

SELECT column1, column2 FROM table1

EXCEPT

SELECT column1, column2 FROM table2;

**INTERSECT Operator:**

Purpose: Returns distinct rows that are common to the result sets of both SELECT statements.

Duplicates: Removes duplicate rows from the result set.

Syntax:

SELECT column1, column2 FROM table1

INTERSECT

SELECT column1, column2 FROM table2;