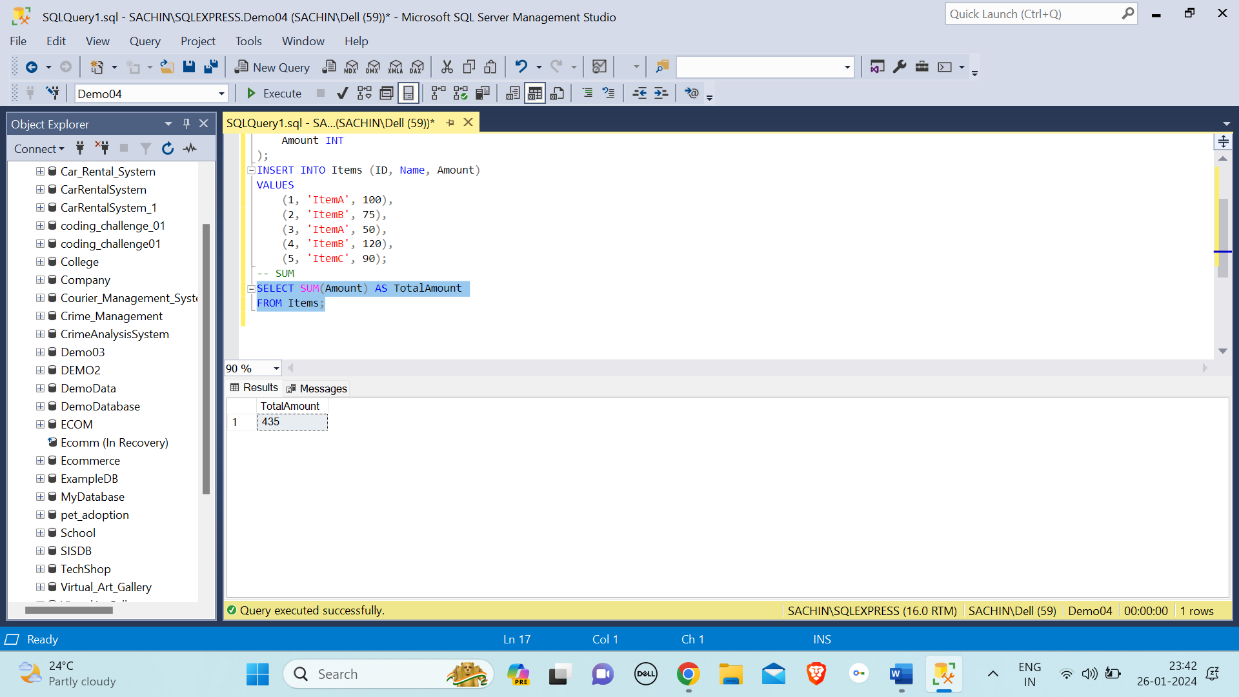
**Total Aggregations using SQL Queries**

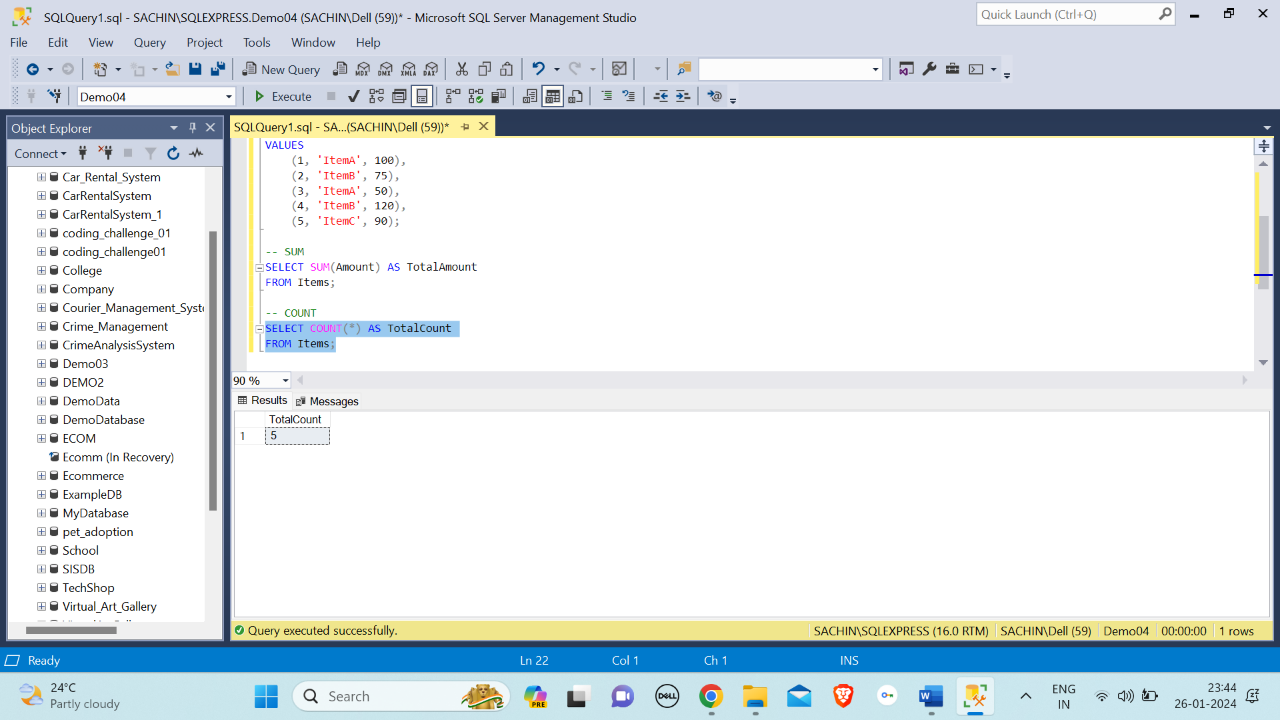
SQL aggregation is the process of collecting a set of values to return a single value. This is done using aggregate functions, such as:

* SUM()
* COUNT()
* AVG()
* GROUP BY
* WITH ROLLUP
* MIN() and MAX()

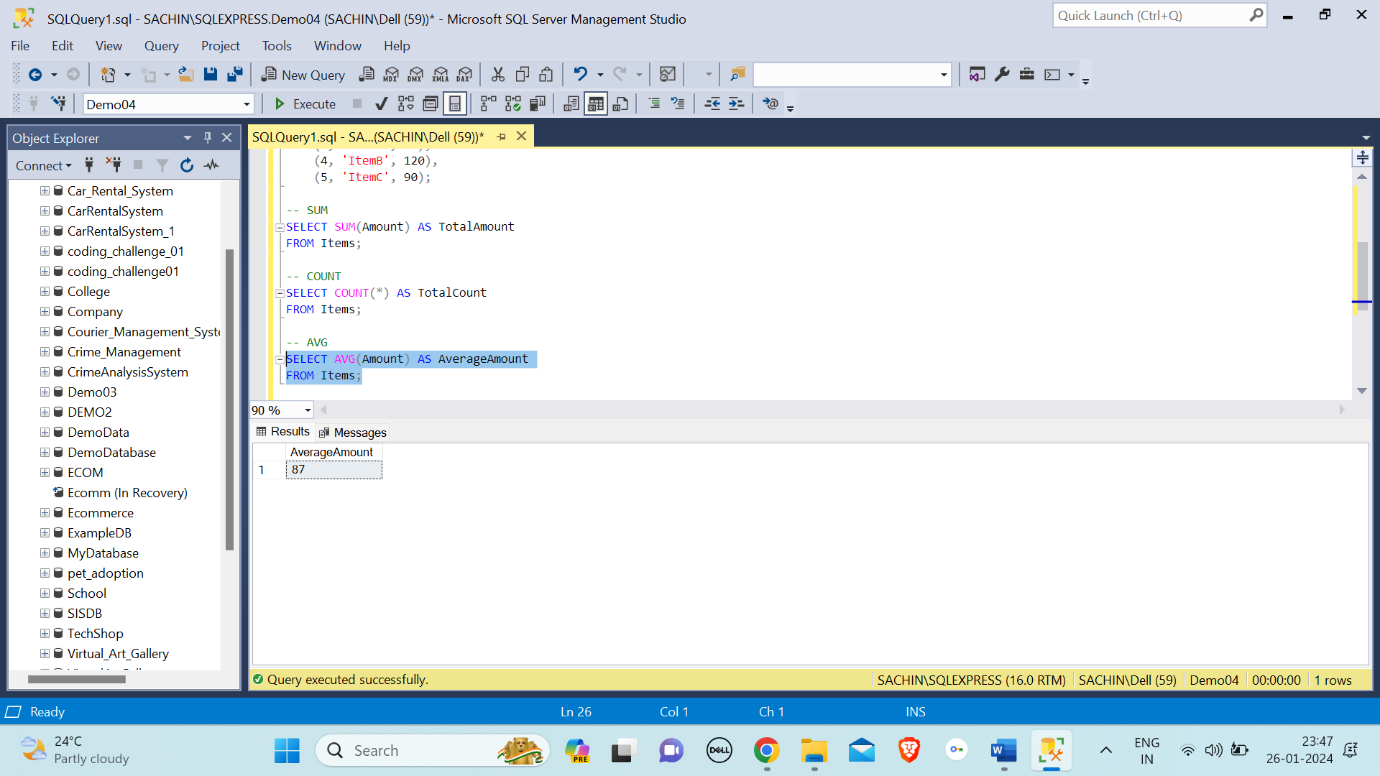
SUM(): Calculates the arithmetic sum of the set of numeric values.



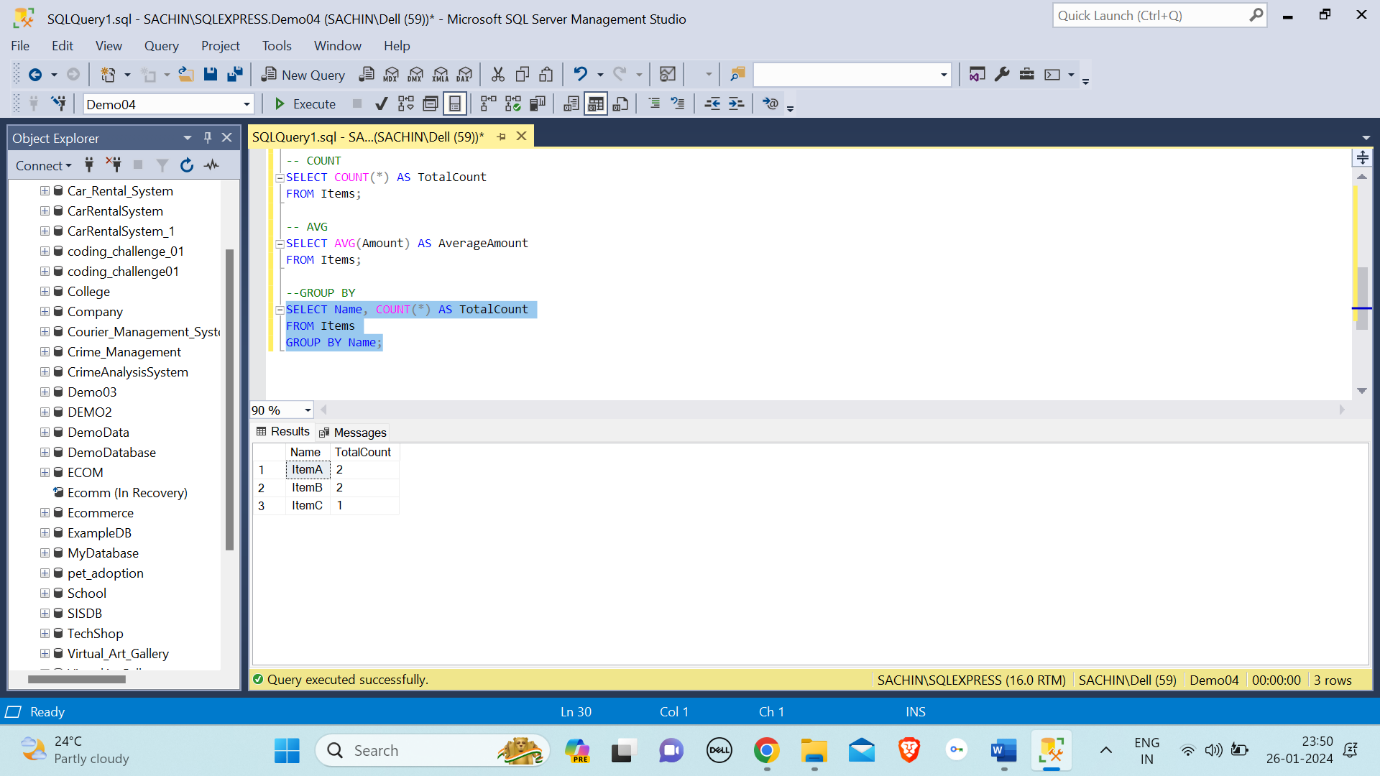
COUNT(): Returns the count of rows. COUNT works on both numeric and non-numeric data types.



AVG(): Calculates the average of a numeric column.

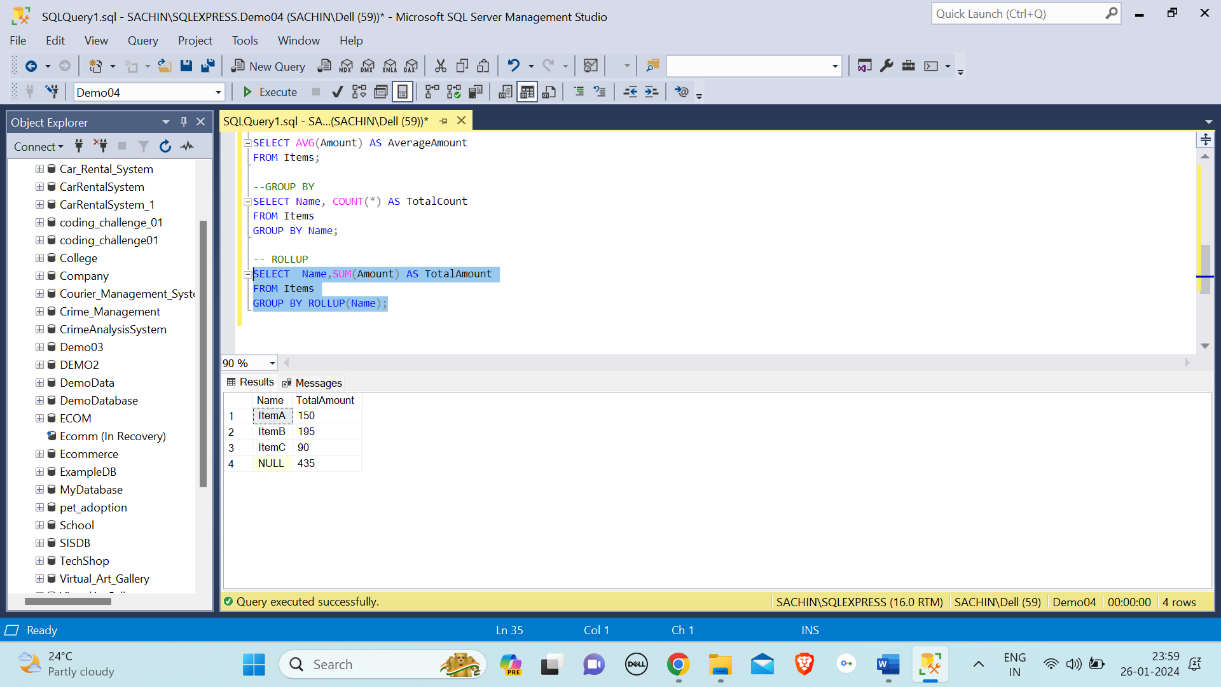


GROUP BY: Allows grouping rows based on certain criteria, and then applying aggregate functions on each group.

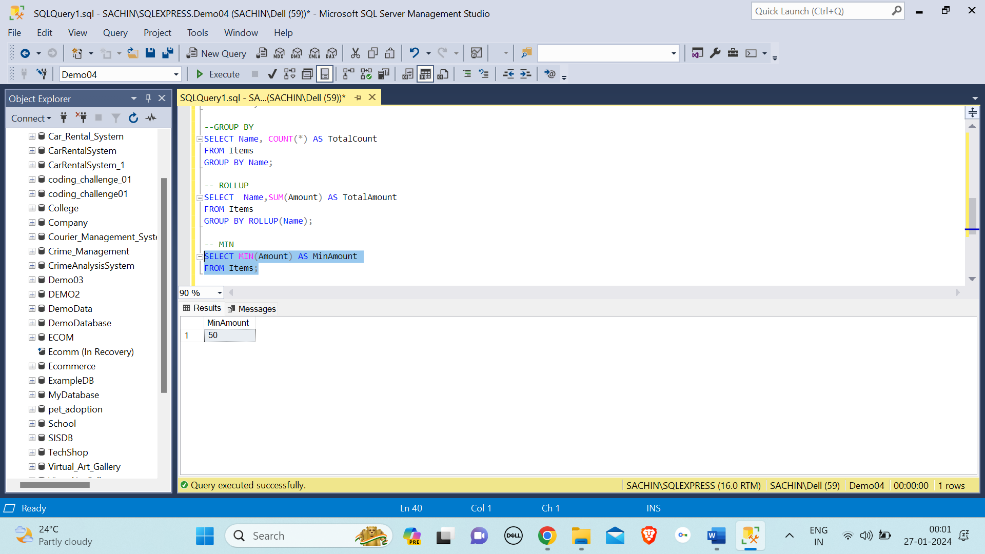


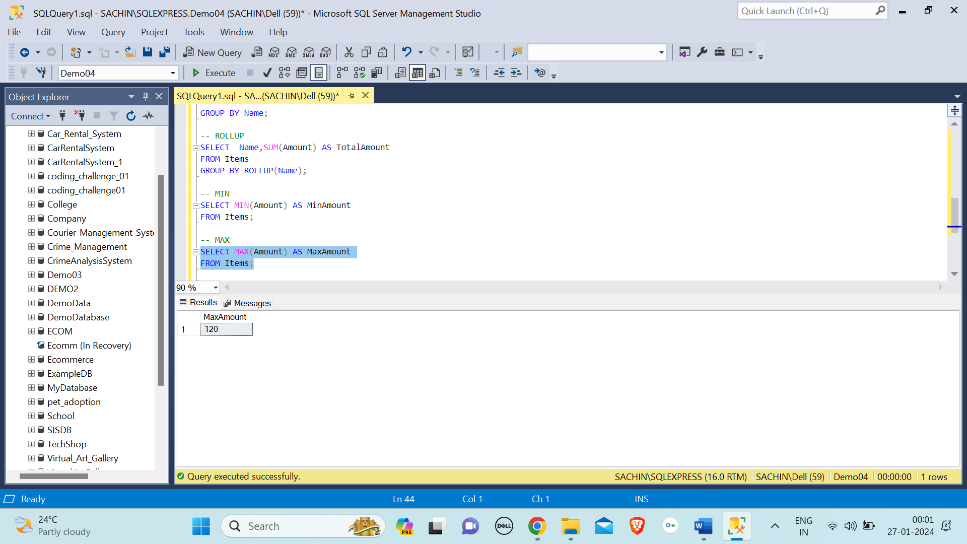
I have used Database Demo04 for total aggregations.

ROLL UP: The ROLLUP operator is used for creating subtotals and grand totals in result sets.



MIN() and MAX(): Find the minimum and maximum values in a column, respectively.



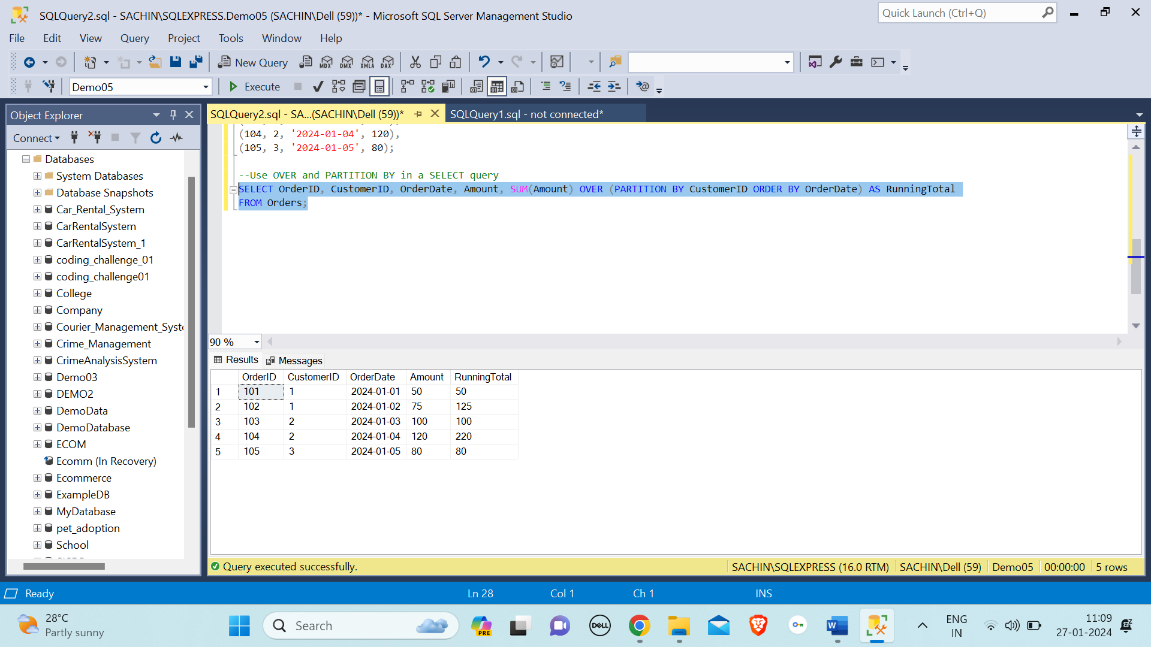


**OVER and PARTITION BY Clause in SQL Queries**

The **OVER** and **PARTITION BY** clauses in SQL are used in combination with window functions to perform calculations across a specified range of rows related to the current row. These clauses are commonly used in analytical queries to generate aggregated results or to perform calculations that involve a window of rows.

OVER Clause: The OVER clause is used with window functions to define the window or set of rows that the function operates on. It allows you to specify the partitioning, ordering, and framing of the window.

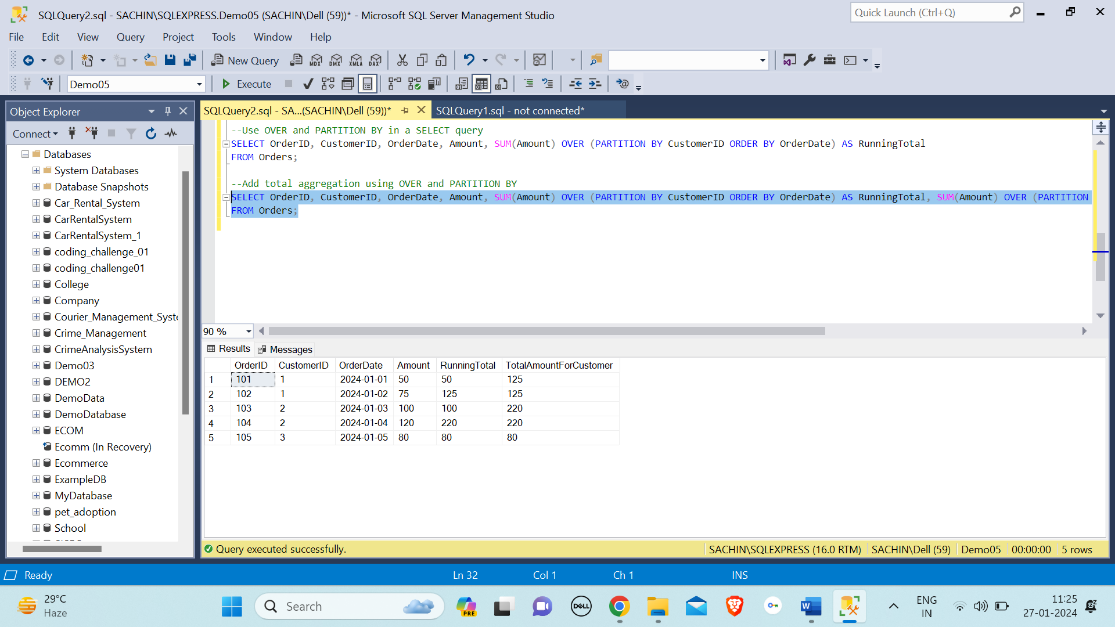
PARTITION BY Clause: the PARTITION BY clause is a subclause of the OVER clause. The PARTITION BY clause splits a large table into smaller partitions. The PARTITION BY clause is useful when performing calculations on individual rows of a group using other rows of that group.



I have created a database named Demo05 and performed over and partition by clause.

**Total Aggregation using OVER and PARTITION BY in SQL Queries**

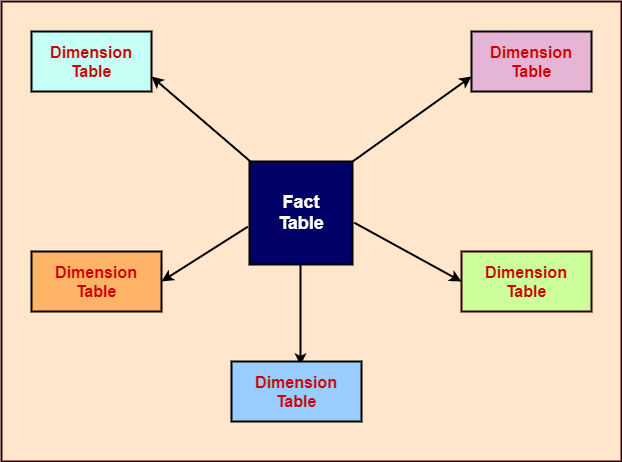
When you want to perform total aggregations using the OVER and PARTITION BY clauses in SQL queries, you are likely interested in calculating cumulative or running totals within each partition.



**Star schemas and Snowflaking schemas**:

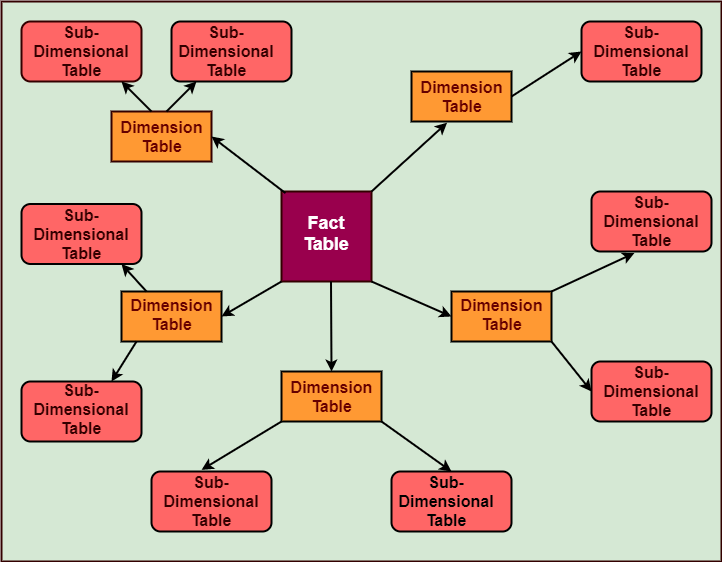
Star Schemas:

In a star schema, the fact table will be at the centre and is connected to the dimension tables. Data redundancy is high and occupies more disk space. The tables are completely in a denormalized structure.



Snowflake Schema:

A snowflake schema is an extension of star schema where the dimension tables are connected to one or more dimensions. Data redundancy is low and occupies less disk space when compared to star schema. The tables are partially denormalized in structure.



**Rules and Restrictions to Group and Filter Data in SQL queries**

In SQL queries, grouping and filtering data are common operations that help organize and extract specific information from a dataset. Here are some rules and restrictions to consider when using these operations:

Grouping Data:

* Aggregation Functions with GROUP BY
* Non-Aggregated Columns

Filtering Data:

* WHERE Clause
* Logical Operators
* IN and NOT IN
* NULL Values
* Comparison Operators
* LIKE Operators

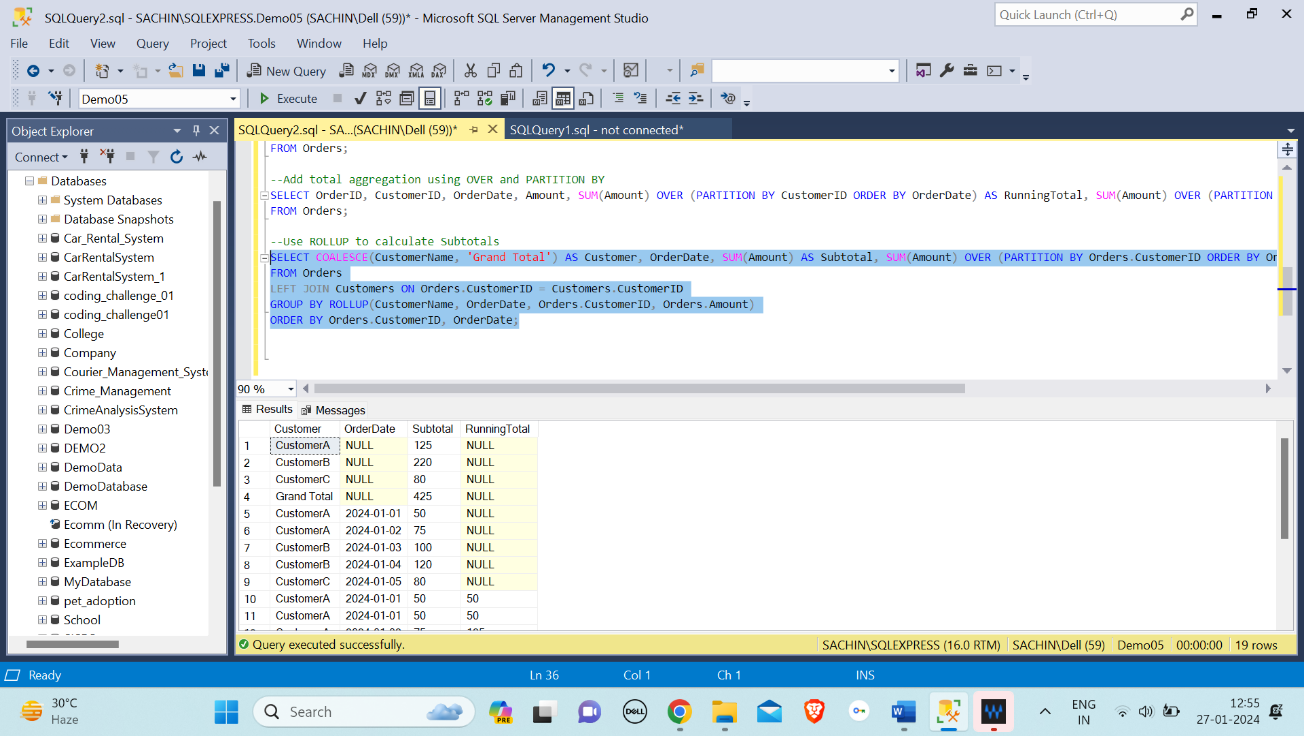
**Order of Execution of SQL Queries**

The order of execution of SQL queries is crucial to understanding how the database processes and retrieves data. SQL queries typically go through several logical processing phases, and understanding this order can be beneficial for optimizing queries and troubleshooting performance issues. The order of execution for a SQL query is:

* FROM
* WHERE
* GROUP BY
* HAVING
* SELECT
* DISTINCT
* ORDER BY
* LIMIT

**How to calculate Subtotals in SQL Queries**

To calculate subtotals in SQL queries, you can use the ROLLUP extension of the GROUP BY statement. This extension creates hierarchical subtotal rows based on its input columns and adds a grand total row to the result set.



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

In SQL Server, the UNION, EXCEPT, and INTERSECT operators are used to combine or compare the results of two or more queries. Here are the key differences between them:

UNION Operator:

Combines the results of two or more SELECT statements into a single result set.

Removes duplicate rows from the combined result set.

EXCEPT Operator:

Returns the rows that are unique to the first SELECT statement and not present in the second SELECT statement.

Removes duplicate rows from the result set.

INTERSECT Operator:

Returns the common rows between two SELECT statements.

Removes duplicate rows from the result set.