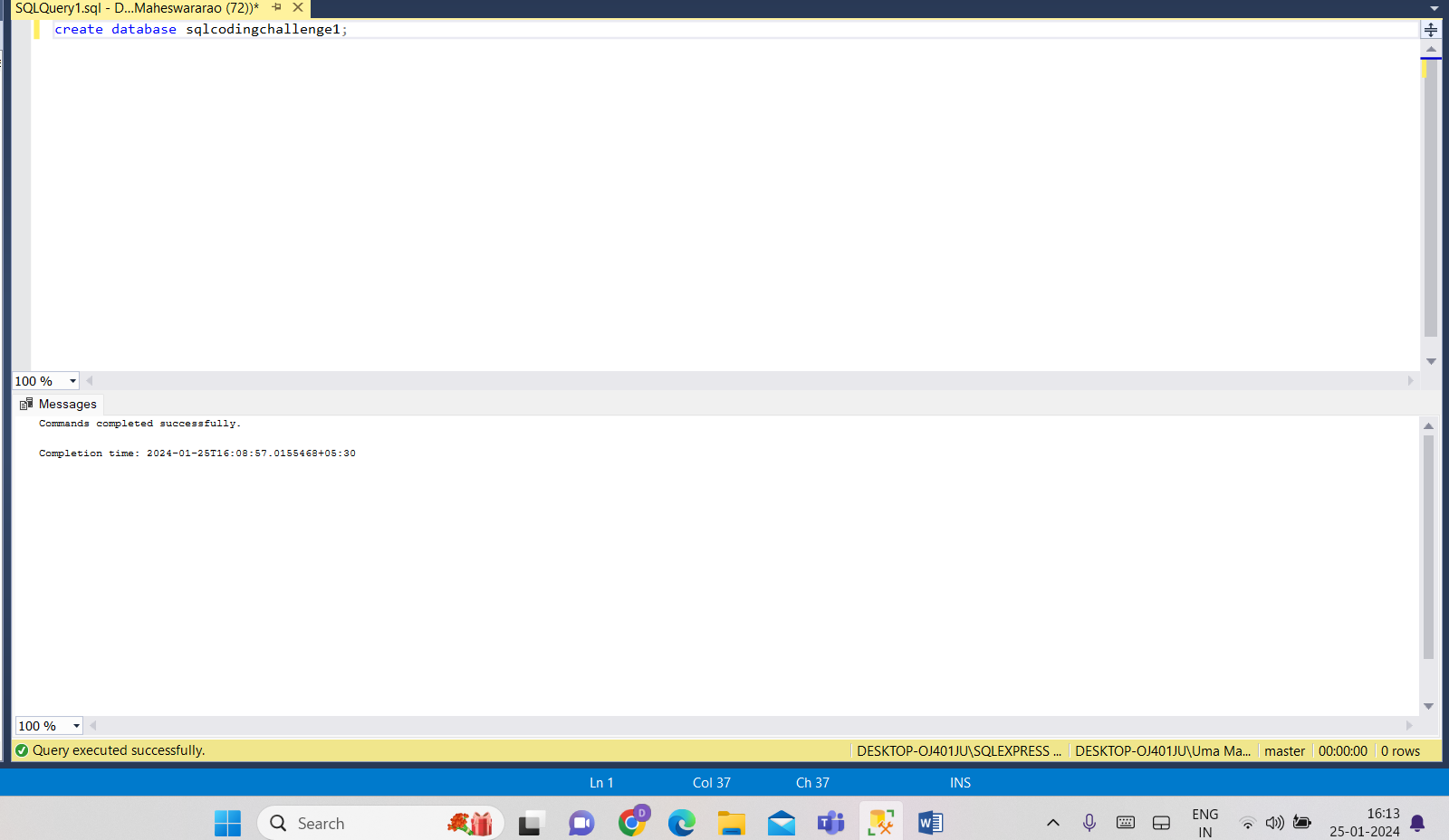
**Name**:Jeevan Sai Badana

Mail: [jeevansai100@gmail.com](mailto:jeevansai100@gmail.com)

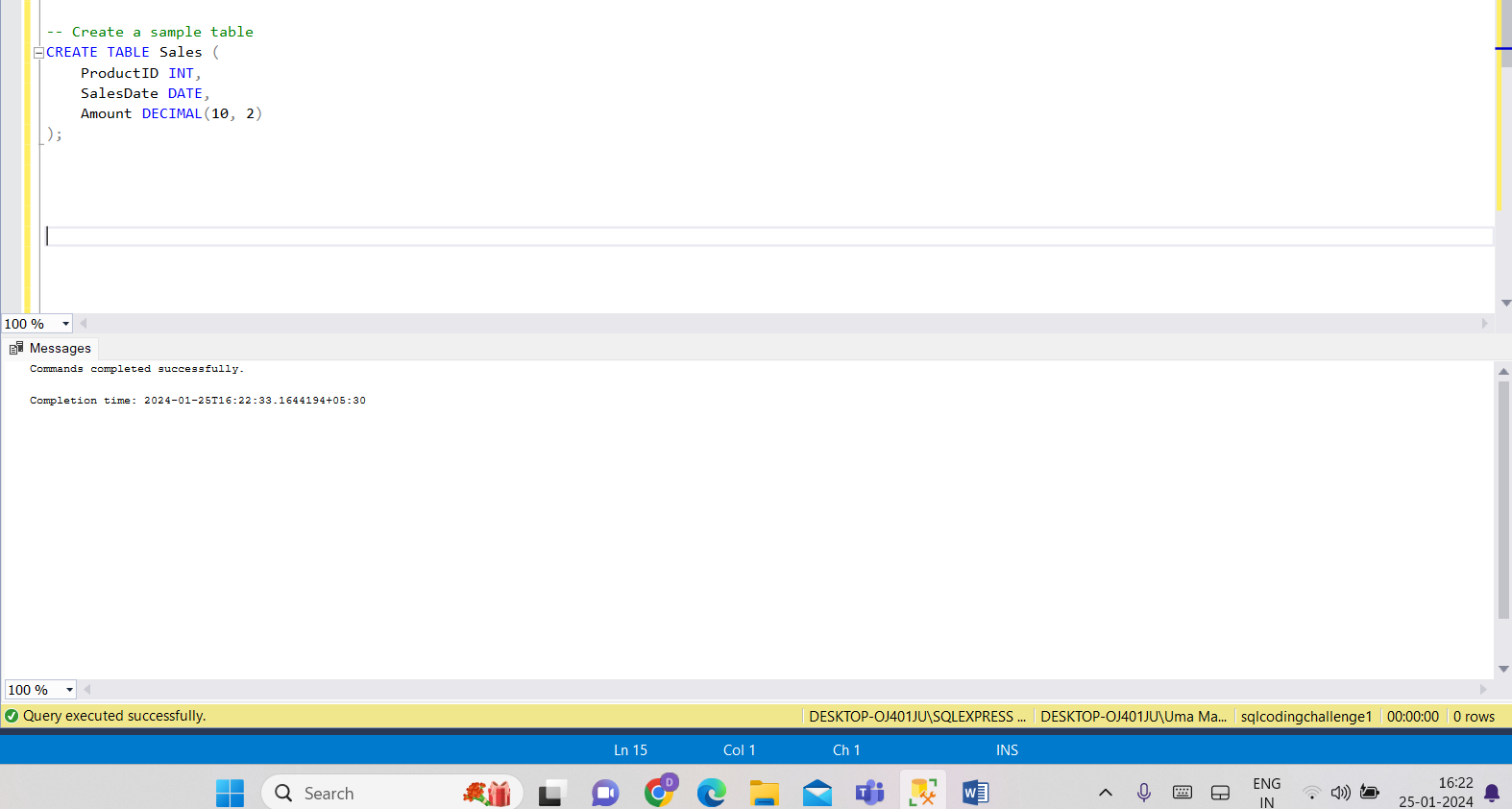
Date : 25-01-2024

1.Execute OVER and PARTITION BY Clause in SQL Queries ,creating subtotals &Total Aggregations using SQL Queries

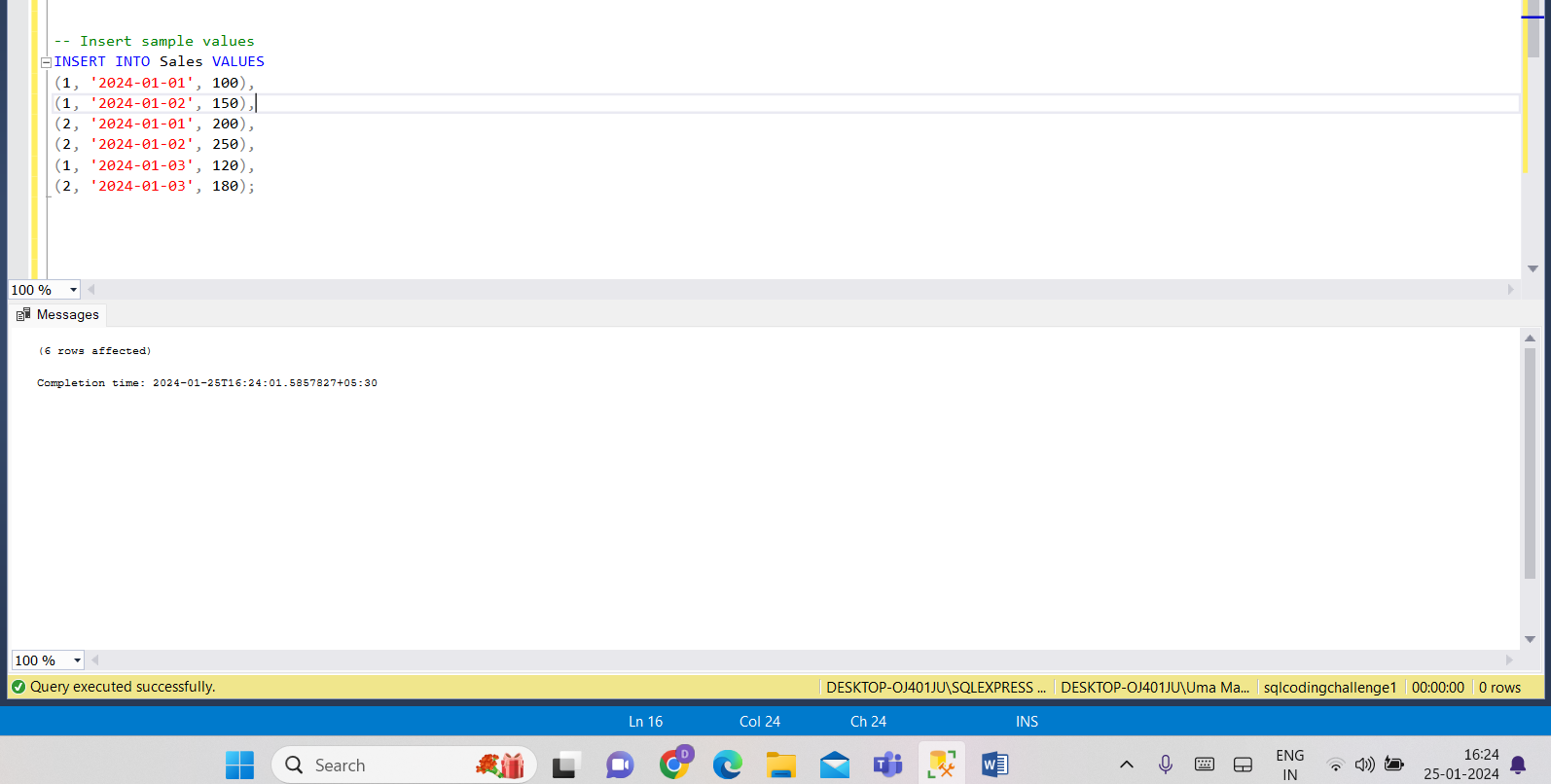
For this first we have to create sqlcodingchallenge1 data base



* Now we create table called Sales



* Now we can insert some values into it



**PARTITION BY Clause**:

The PARTITION BY clause is used to divide the result set into partitions to which the window function is applied.

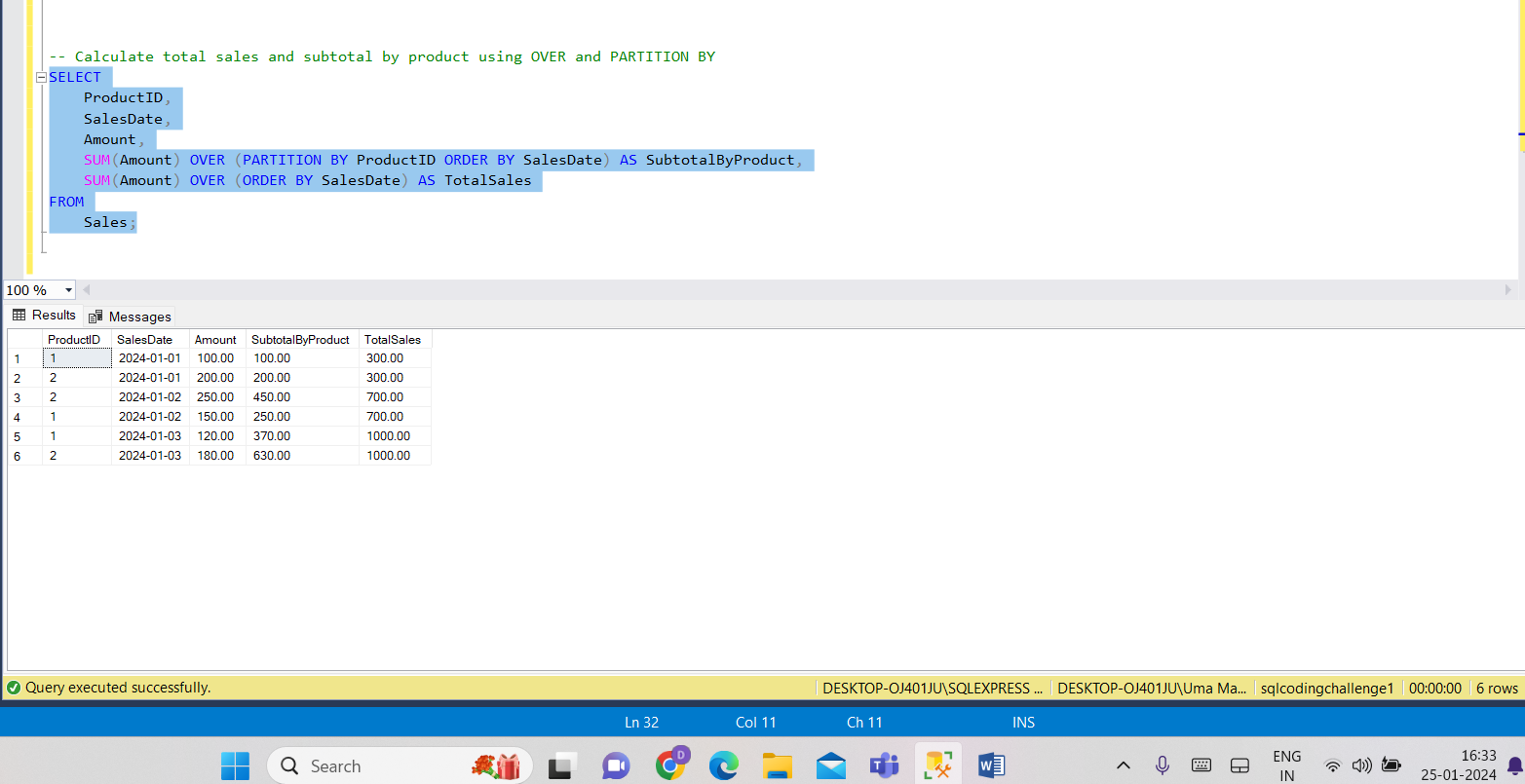
**OVER Clause**:

The OVER clause is used with window functions to define the window or set of rows over which the function operates. It includes the PARTITION BY clause to specify the grouping and can also include ORDER BY to define the order within the partition.

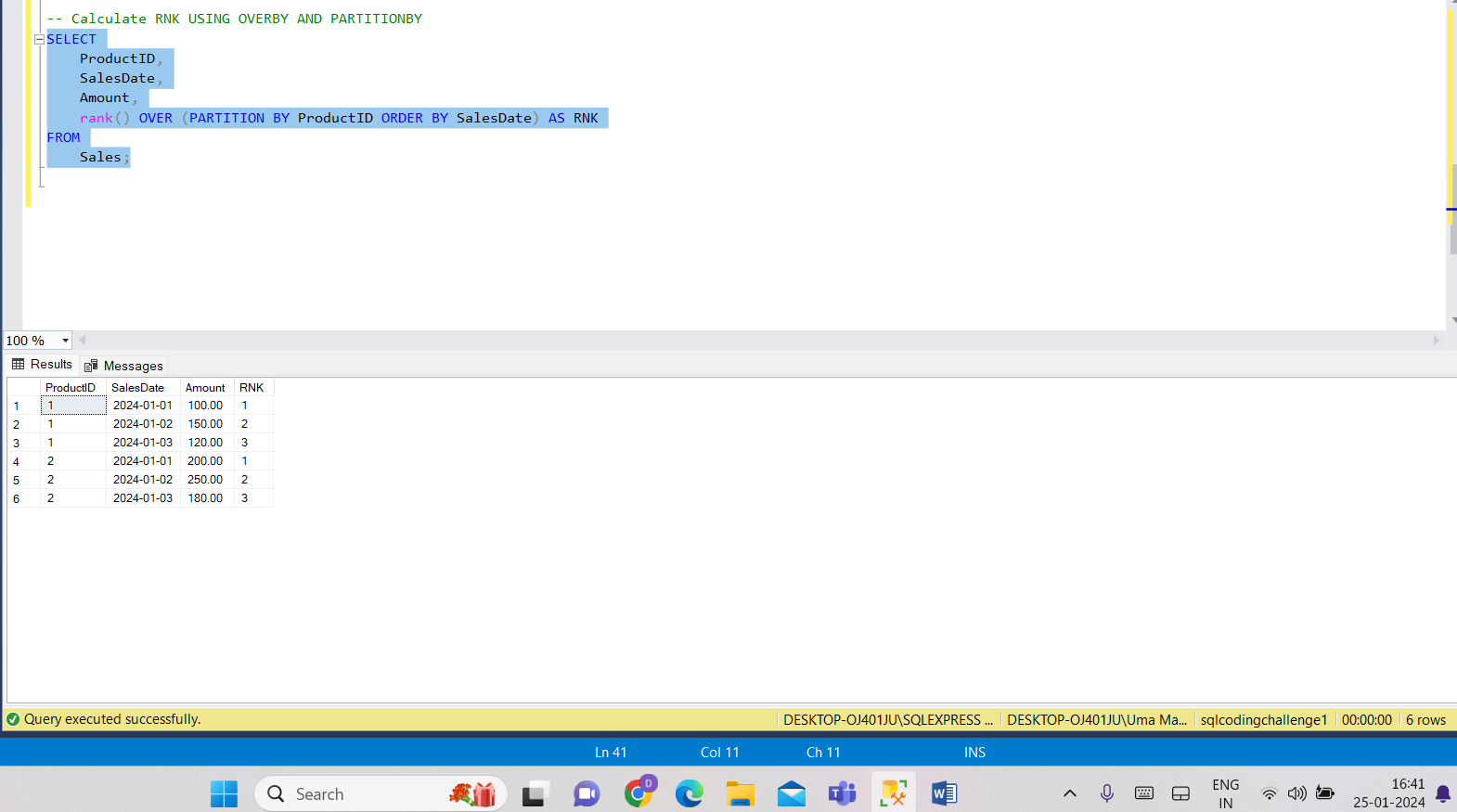
**Creating Subtotals**:

Subtotals can be created by using window functions to calculate aggregates within each partition. This is useful when you want to calculate running totals, averages, or other aggregates for each group in your result set.

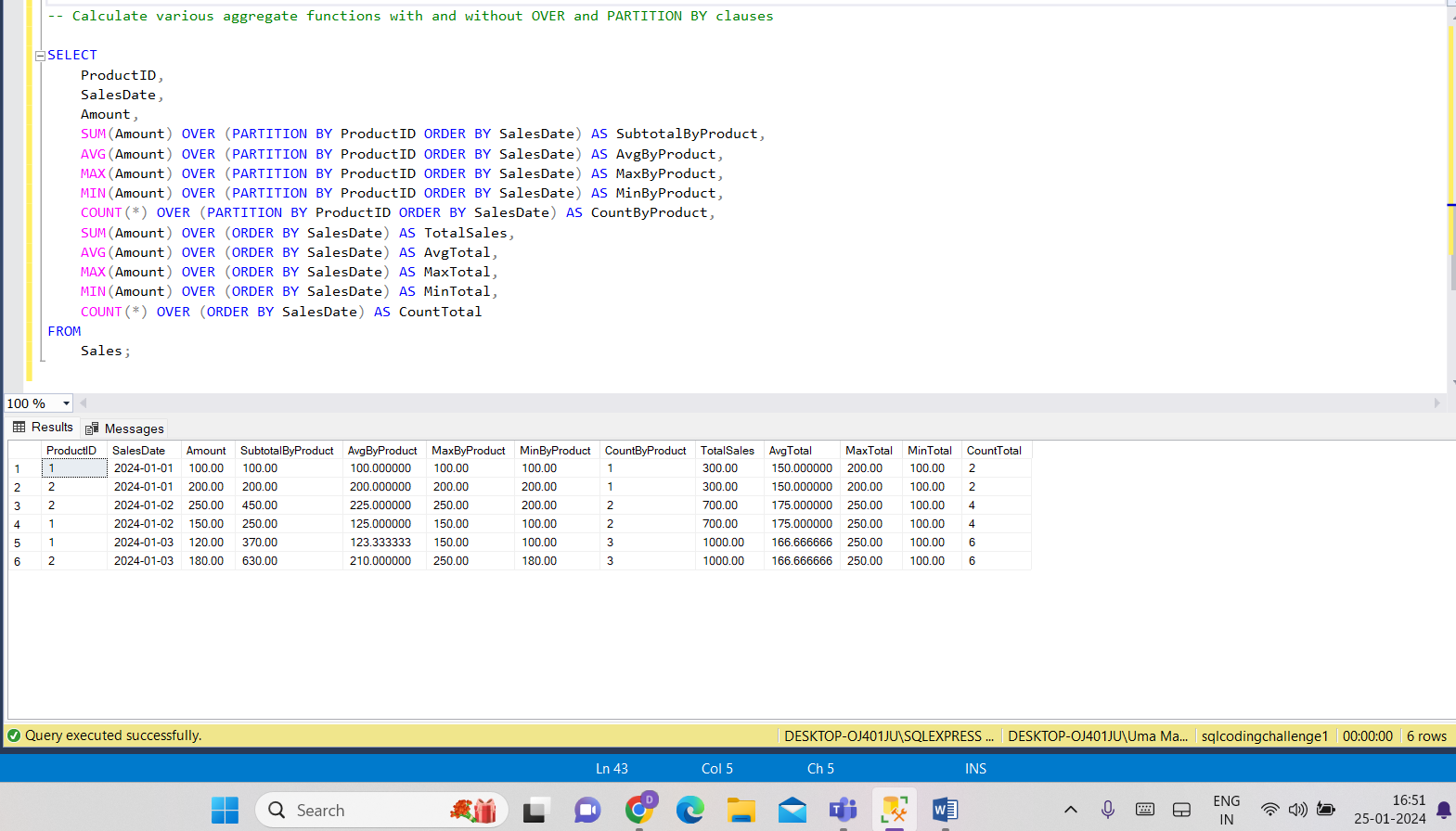
* Calculate total sales and subtotal by product using OVER and PARTITION BY



* Calculate RNK USING OVERBY AND PARTITIONBY

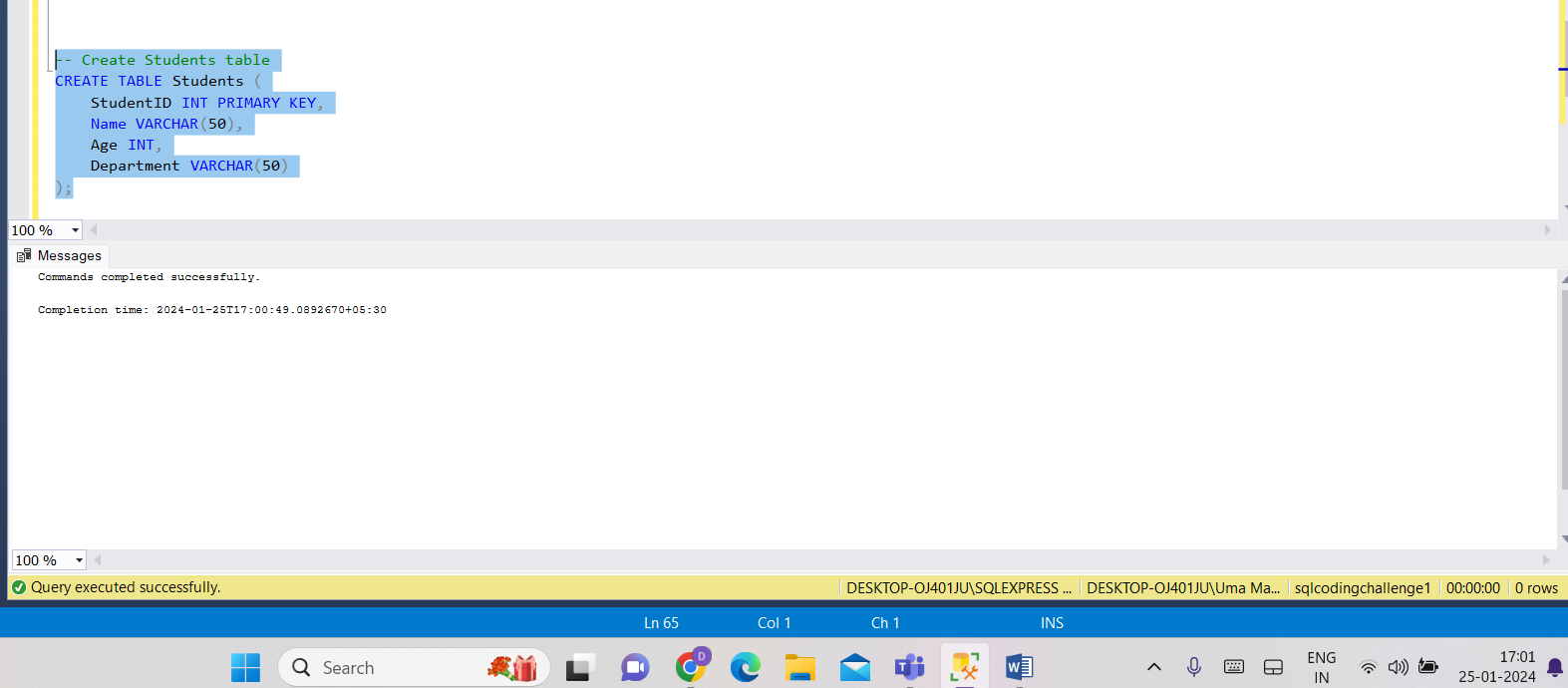


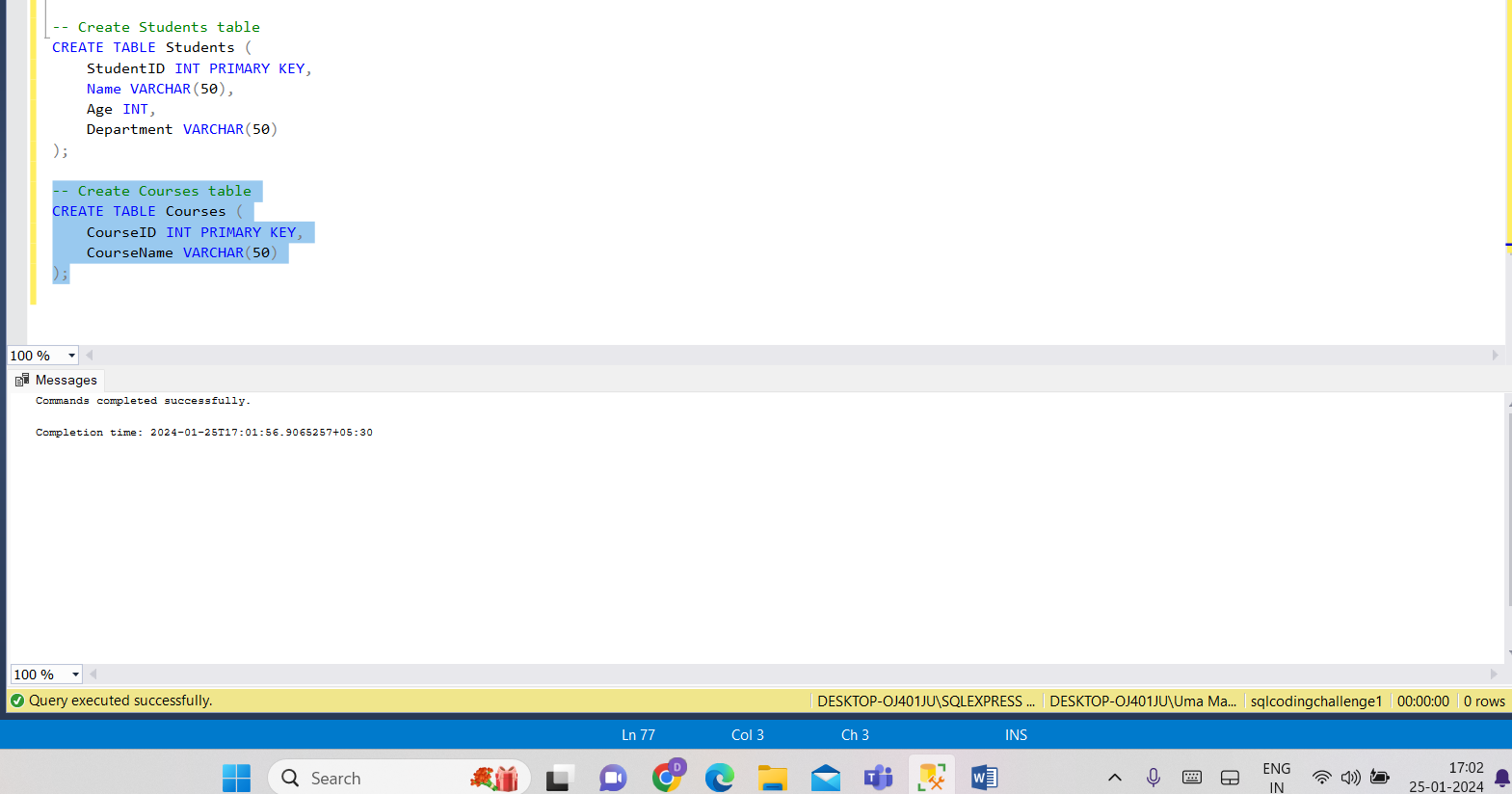
* Now we can calculate various aggregates using OVER and PARTITION BY clauses
* Various Aggregate functions means Sum,Avg,Max,Min,Count
* We calculate aggregate functions both with and without the PARTITION BY clause.



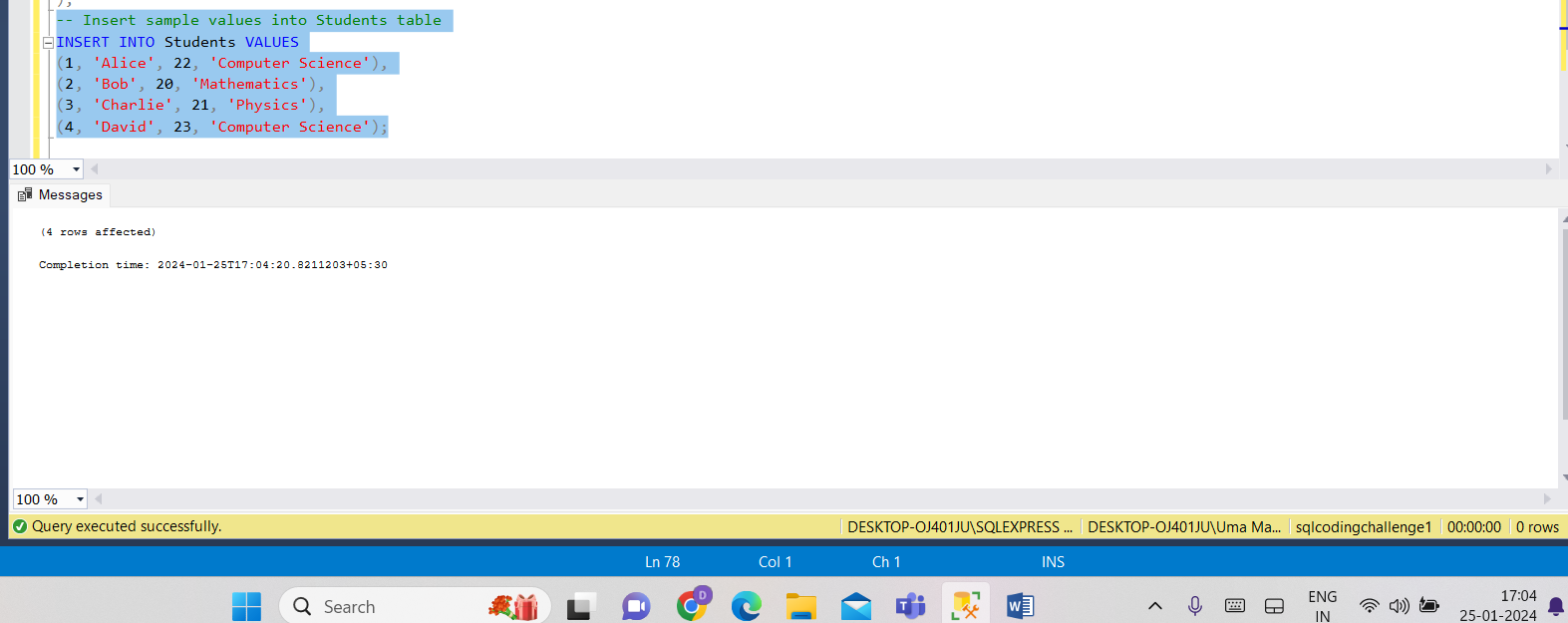
**2. Execute all the join with examples**.

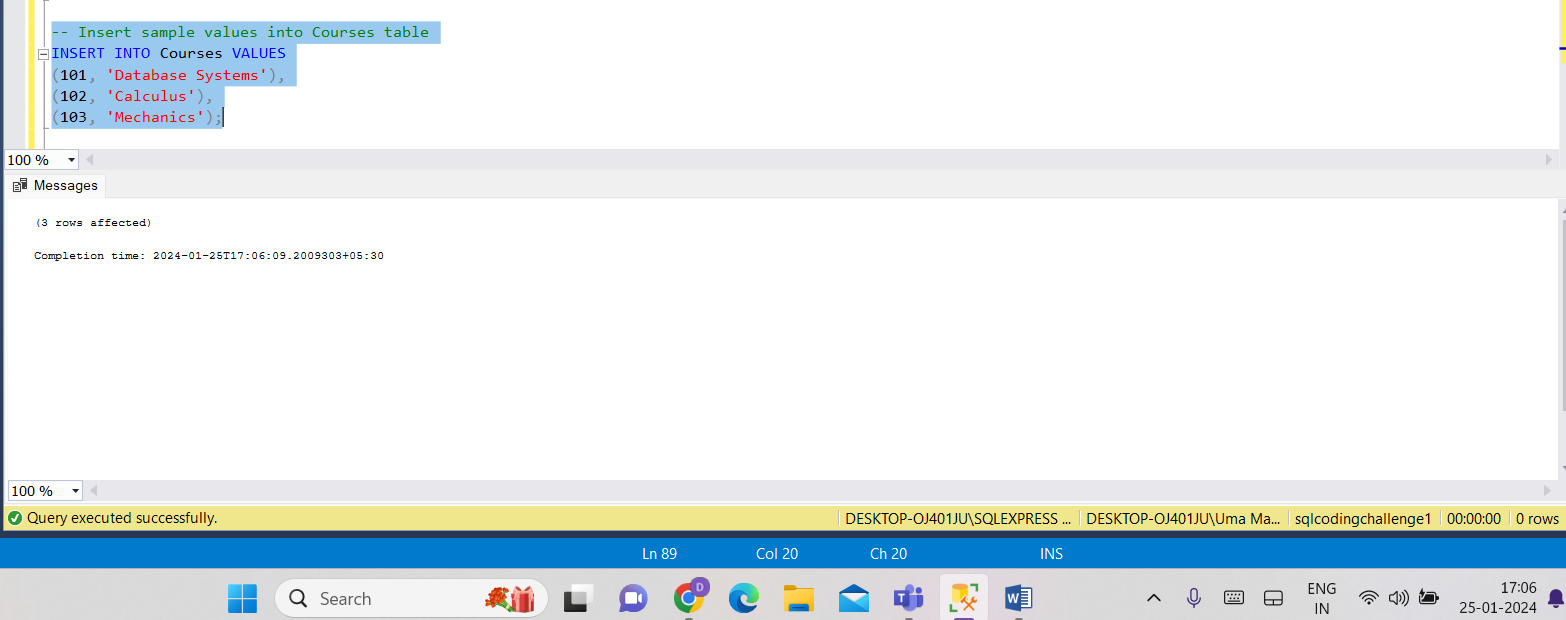
For this we can create two tables students and courses



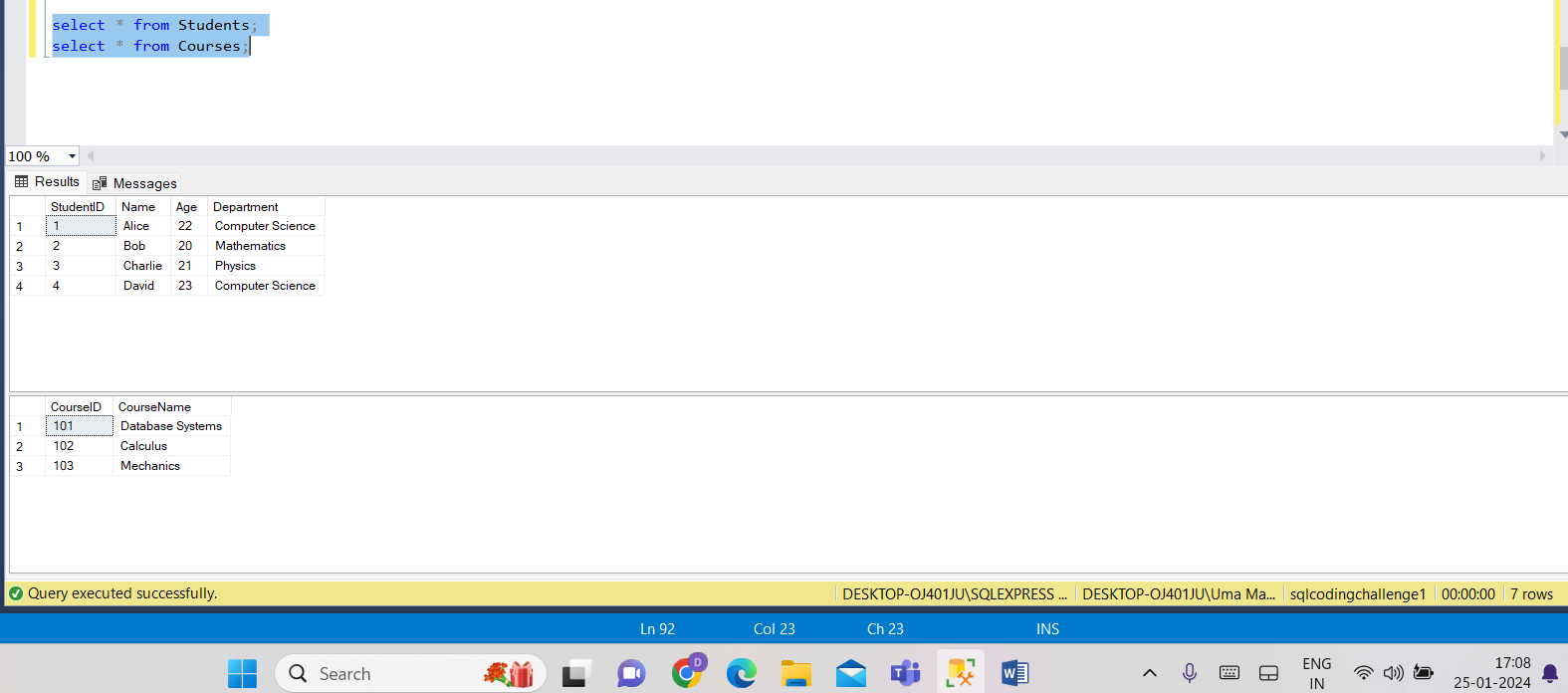


Now insert some random values into two tables





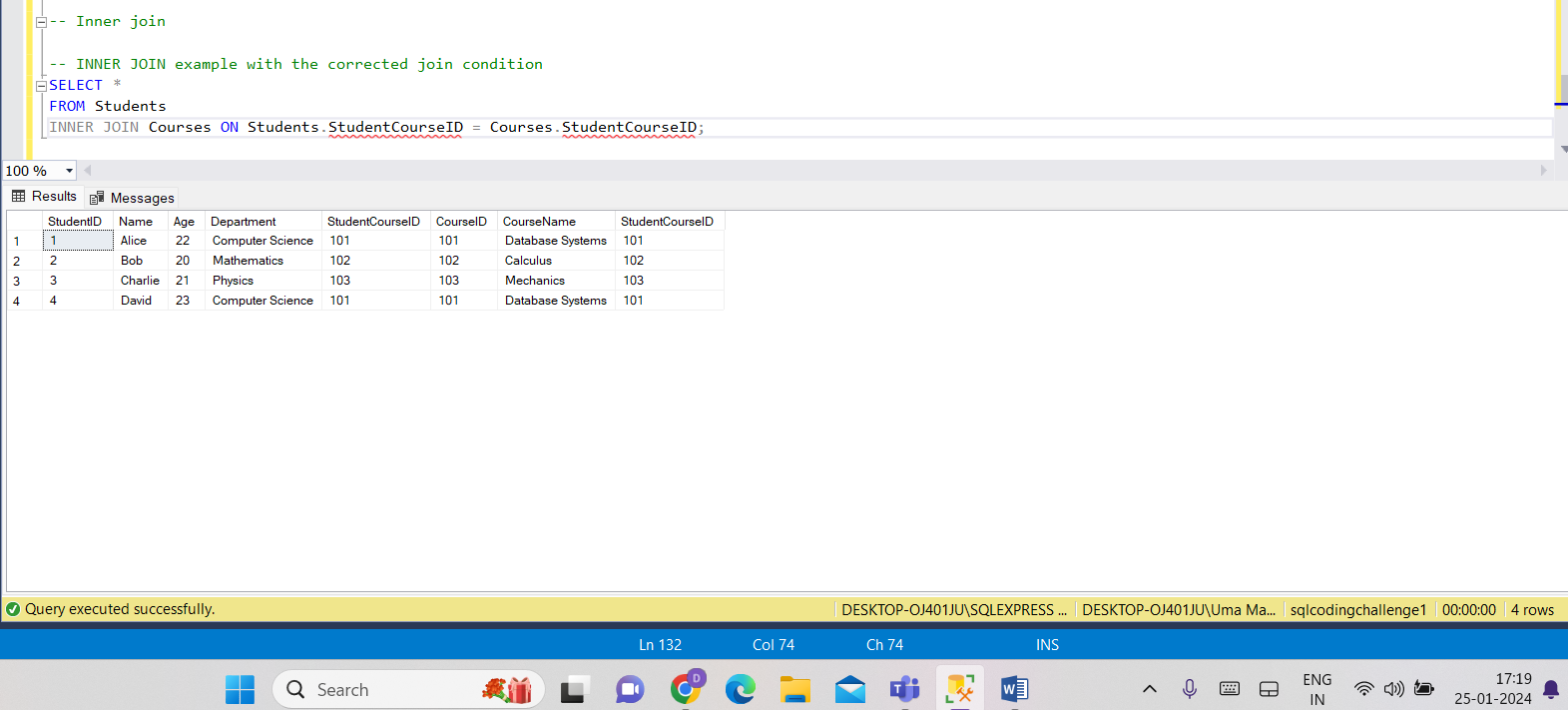
Now we just retrieve everything from these two tables using select statements to check values inserted or not



**INNER JOIN**:

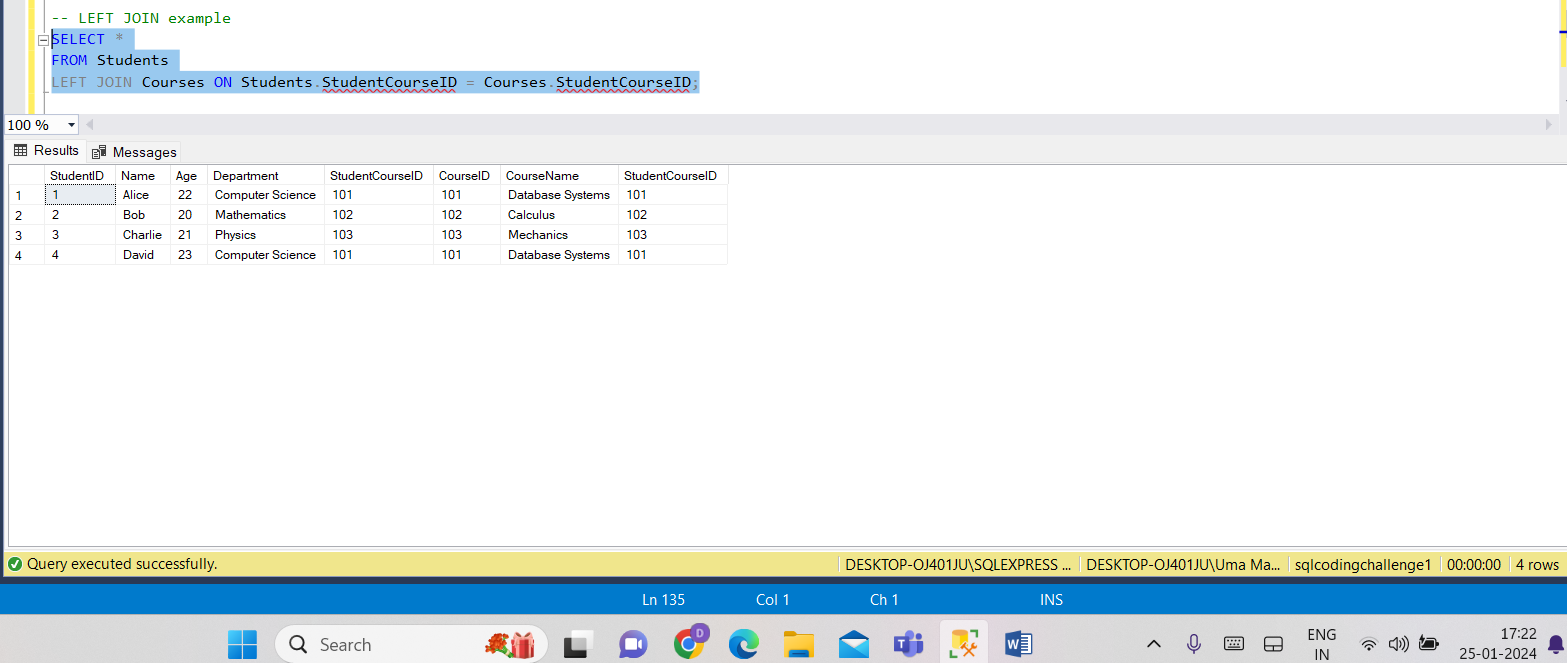
* Returns only the rows where there is a match in both tables based on the specified condition.

I forgot to add same column so I use alter command to add same column in both tables



**LEFT JOIN (or LEFT OUTER JOIN):**

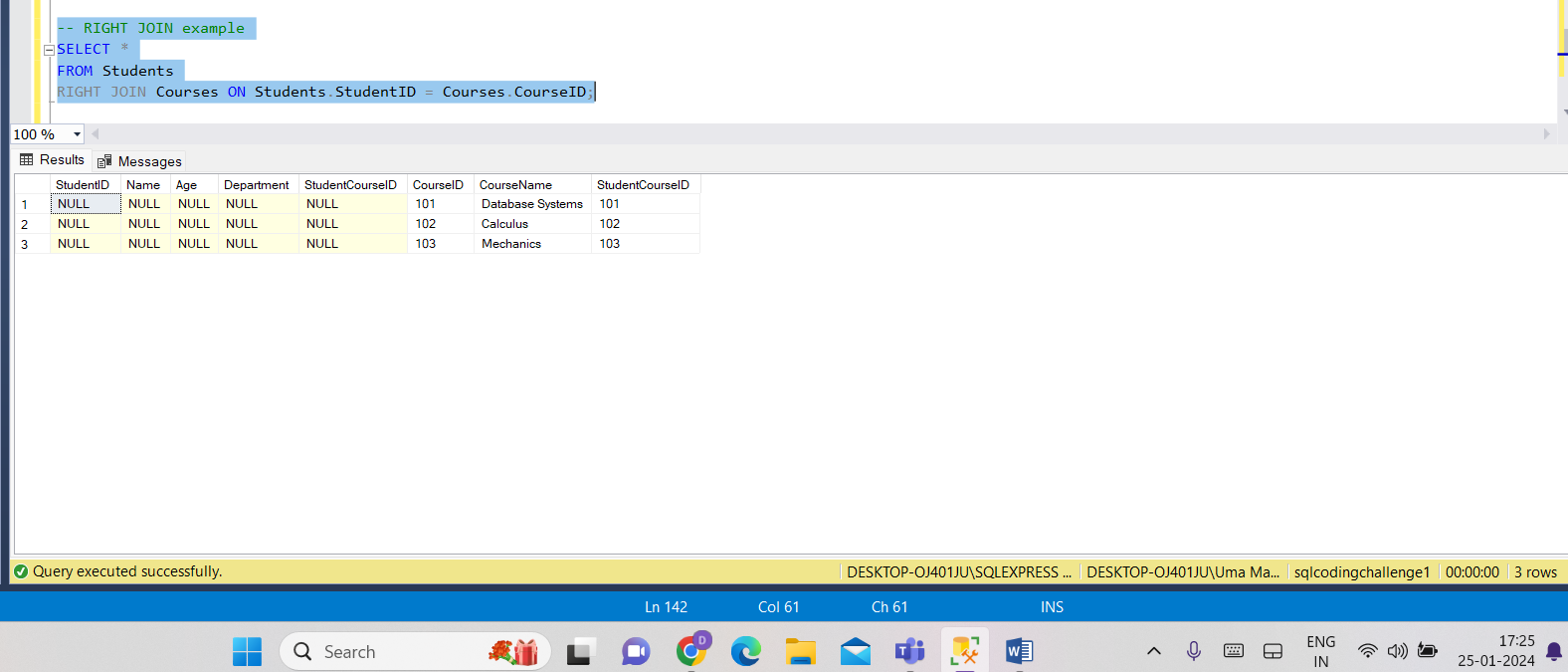
Returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.



**RIGHT JOIN (or RIGHT OUTER JOIN):**

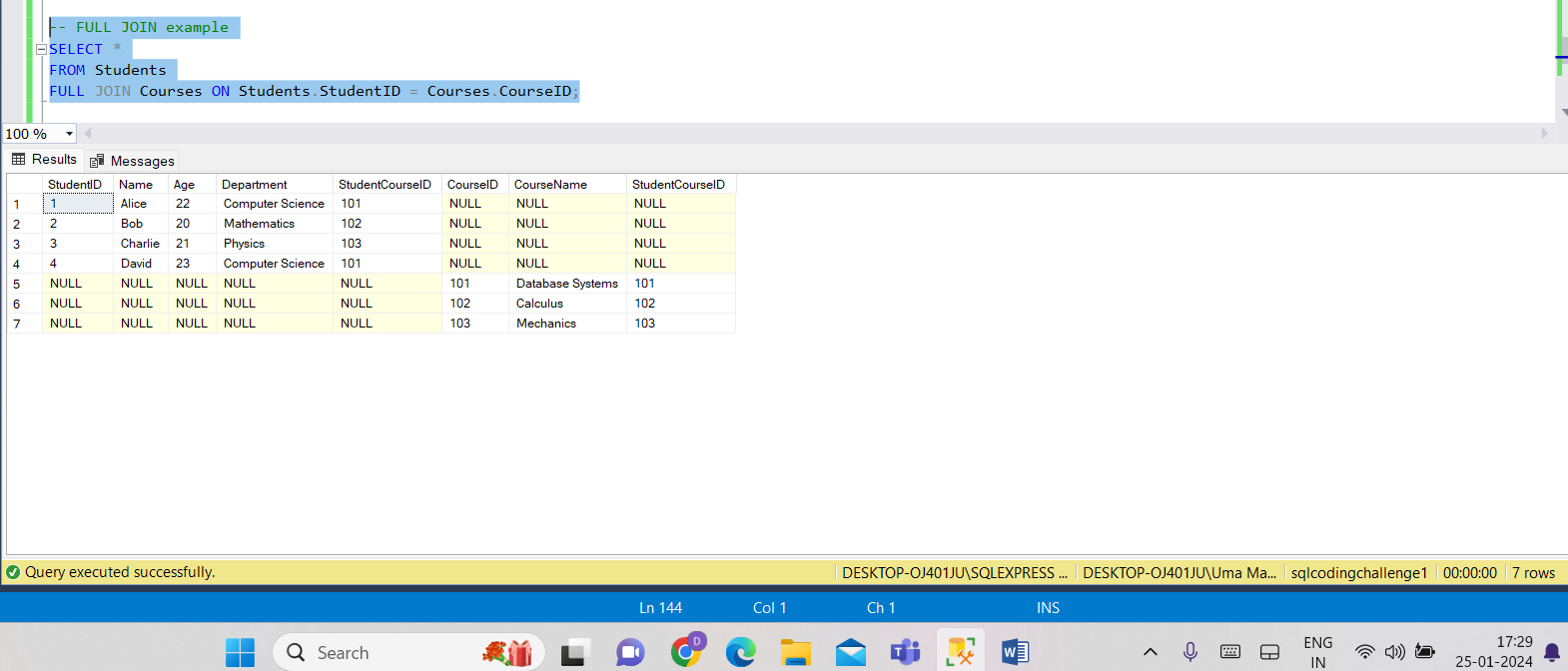
Returns all rows from the right table and the matched rows from the left table. If there is no match, NULL values are returned for columns from the left table.

* Here I want to give course id and student id so both are not same and we get null values in left part of table



**FULL JOIN (or FULL OUTER JOIN):**

Returns all rows when there is a match in either the left or right table. If there is no match, NULL values are returned for columns from the table without a match.



**CROSS JOIN:**

Returns the Cartesian product of both tables, i.e., all possible combinations of rows from both tables.

