SQL CODING CHALLENGE

1) **A) Execute OVER and PARTITION BY Clause in SQL Queries**
   B) creating subtotals
   C) Total Aggregations using SQL Queries.

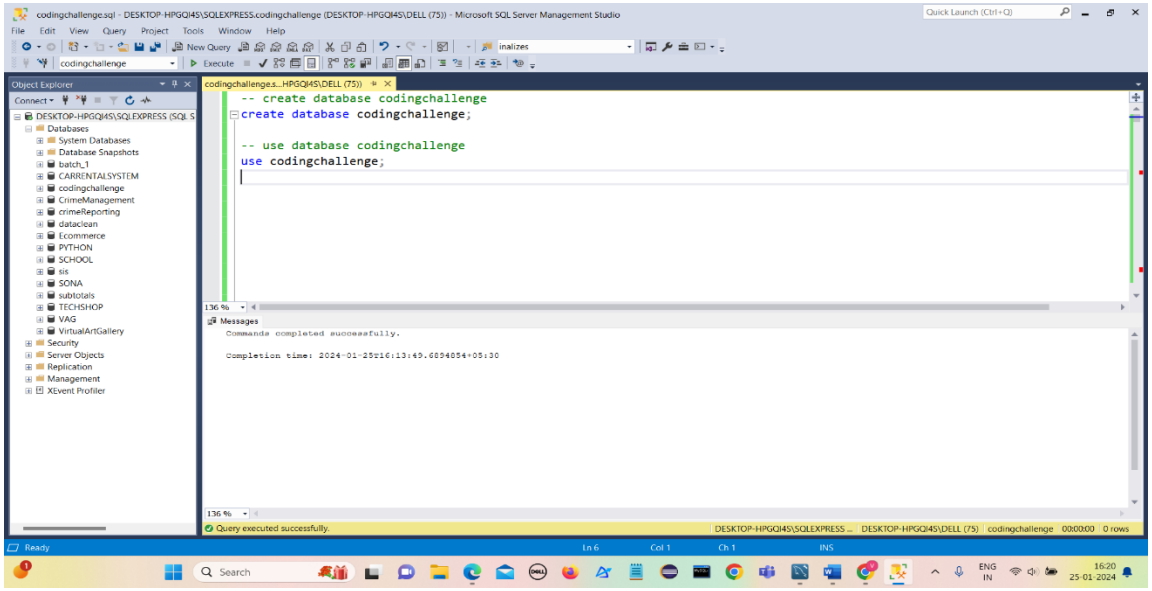
Creating a database:

Create a database named codingchallenge

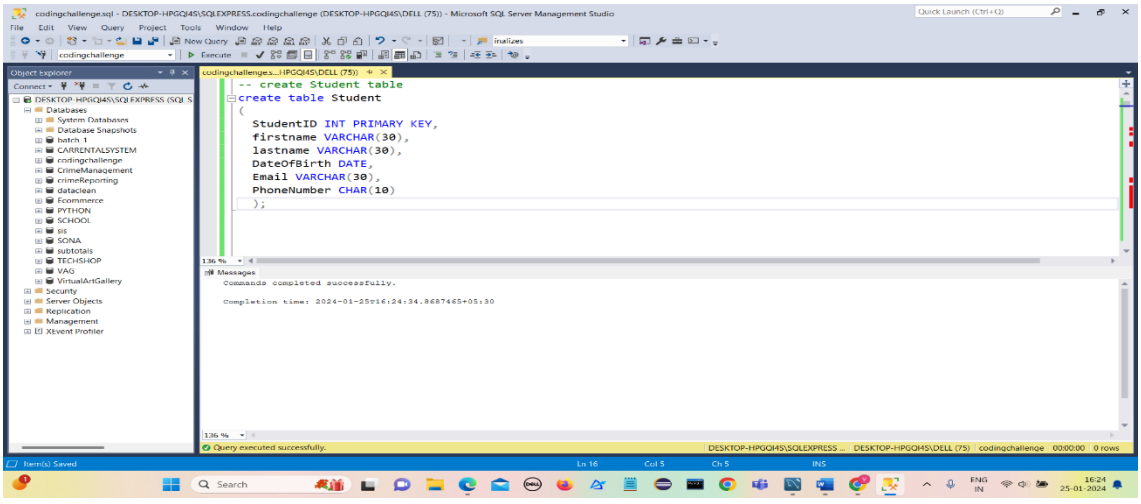
Use the database:

use codingchallenge;

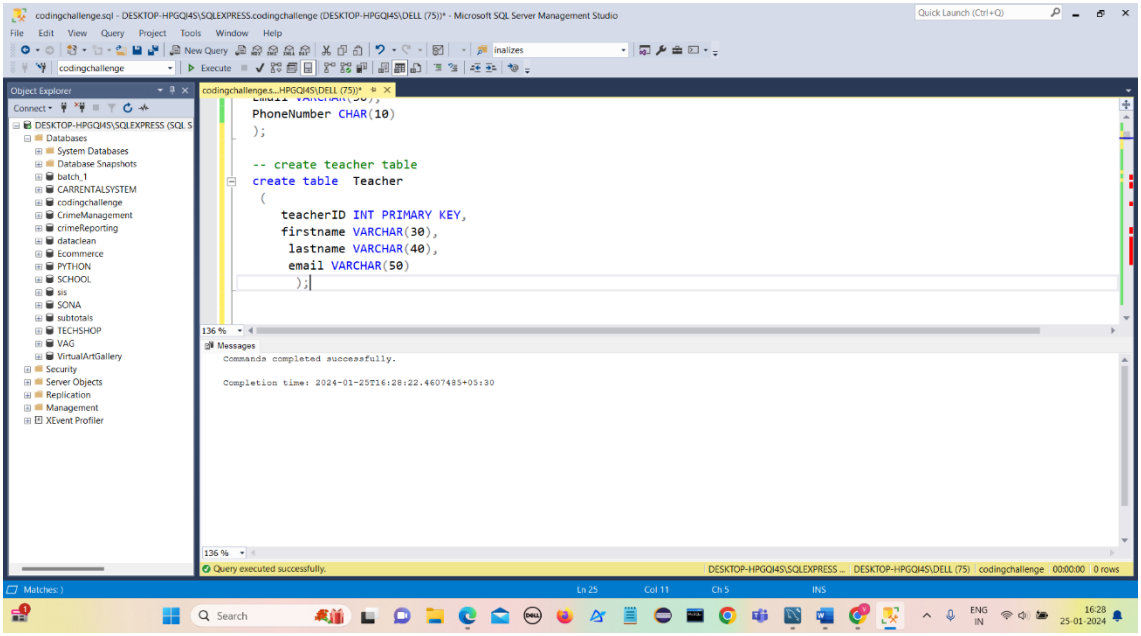
Use the database to make it as the current working database on which different operations can be performed.



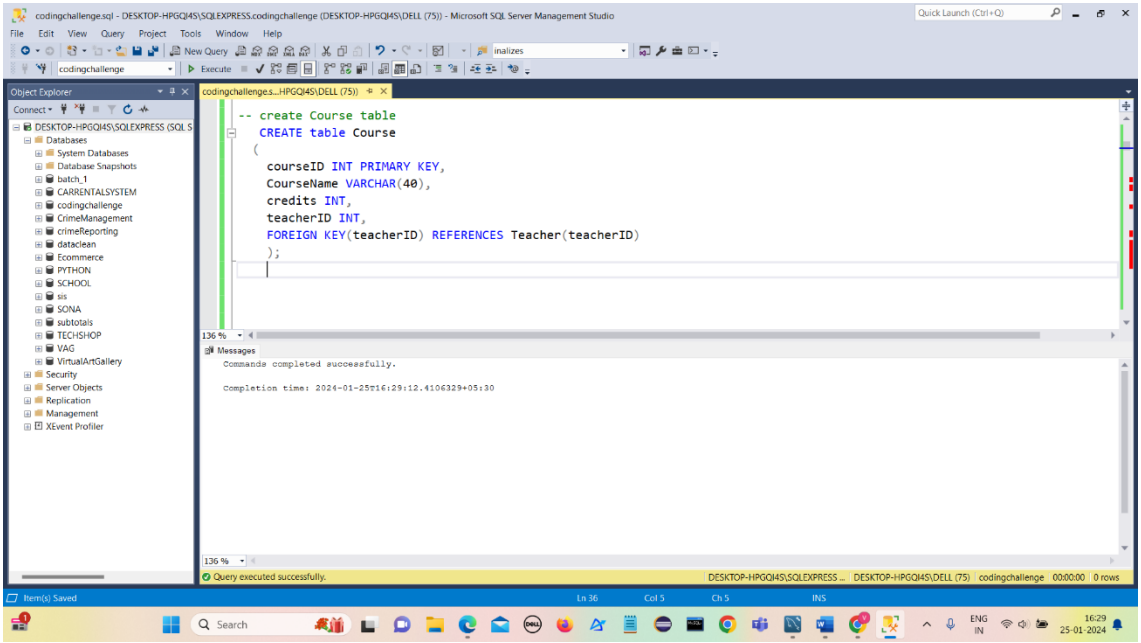
1) Create student table using create statement and give attributes



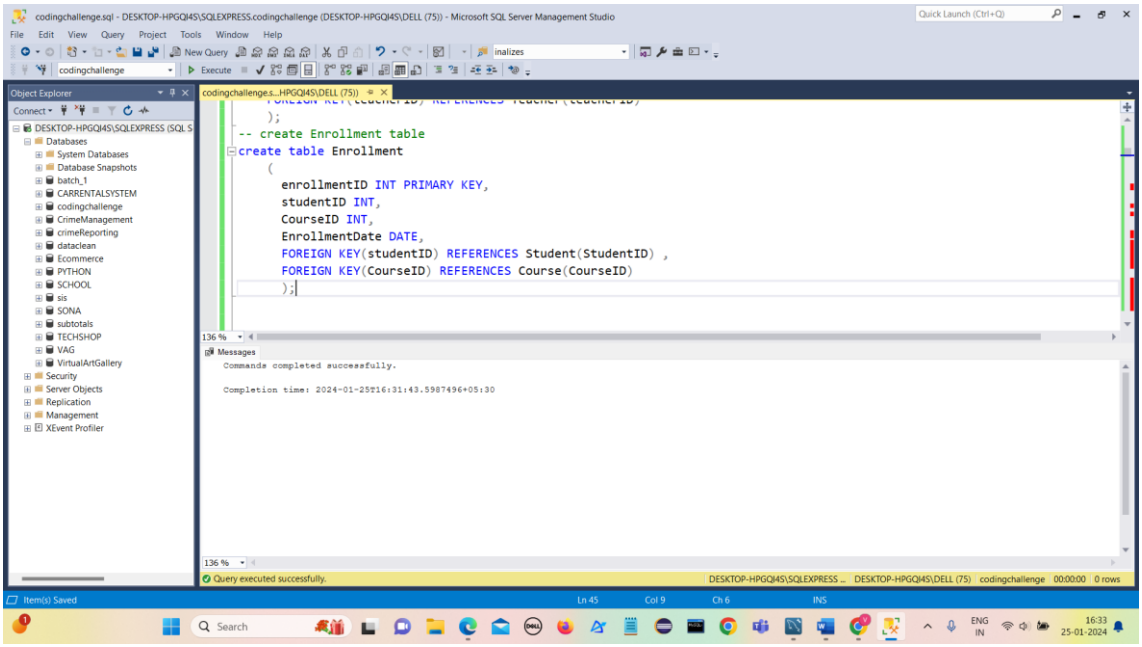
2) Create table Teacher



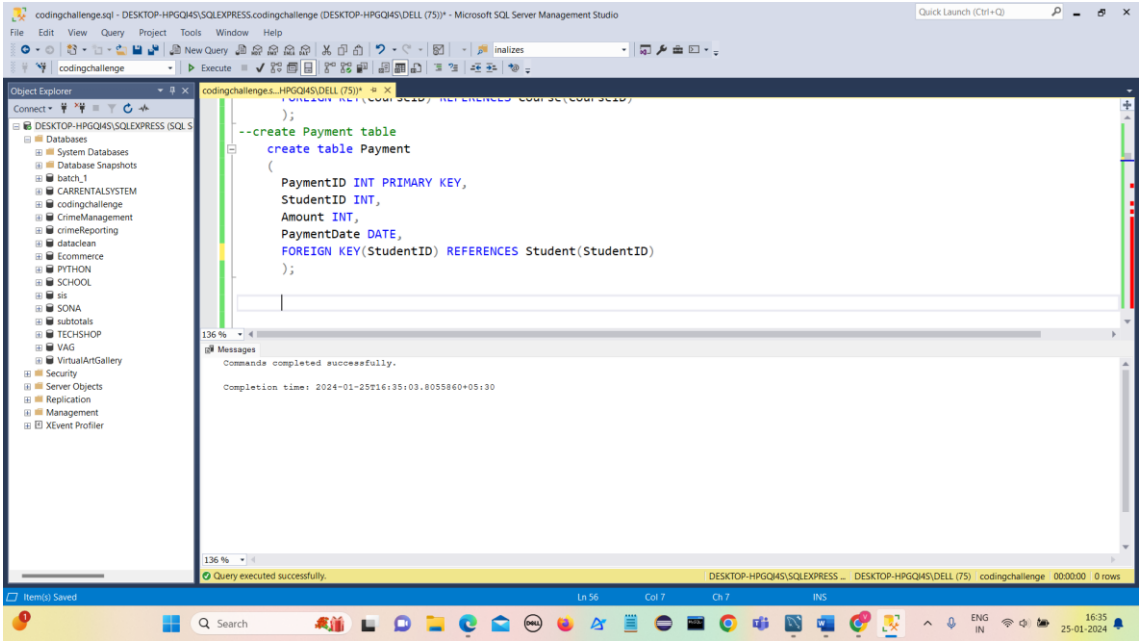
3) Create Course Table



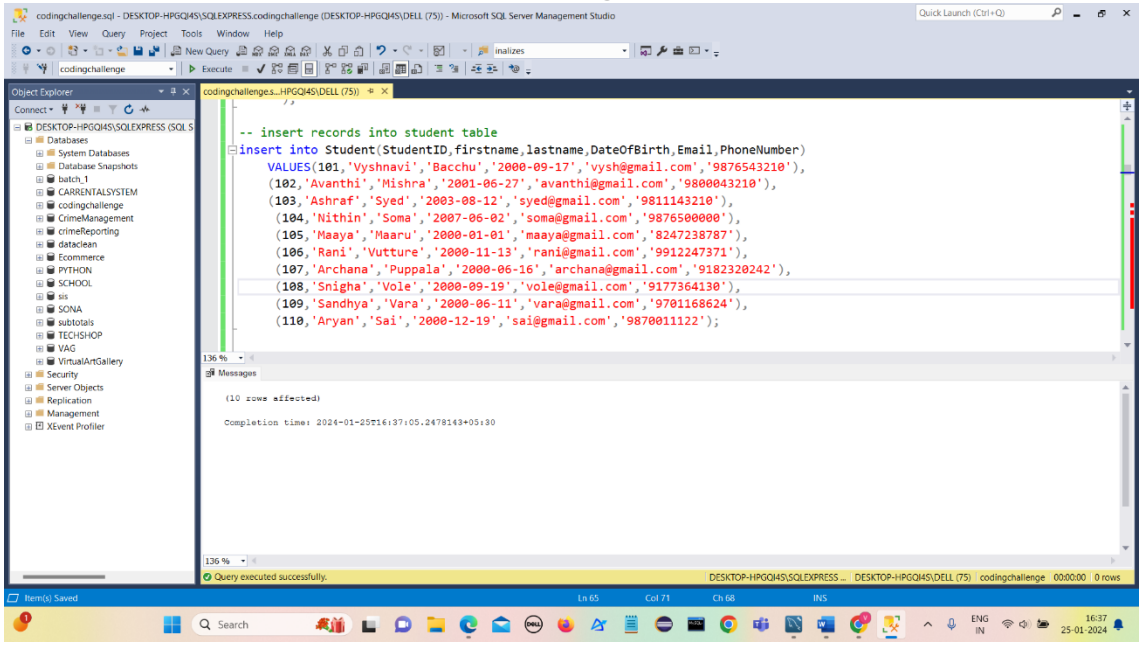
4) Create Enrollment Table



5) Create Payment table

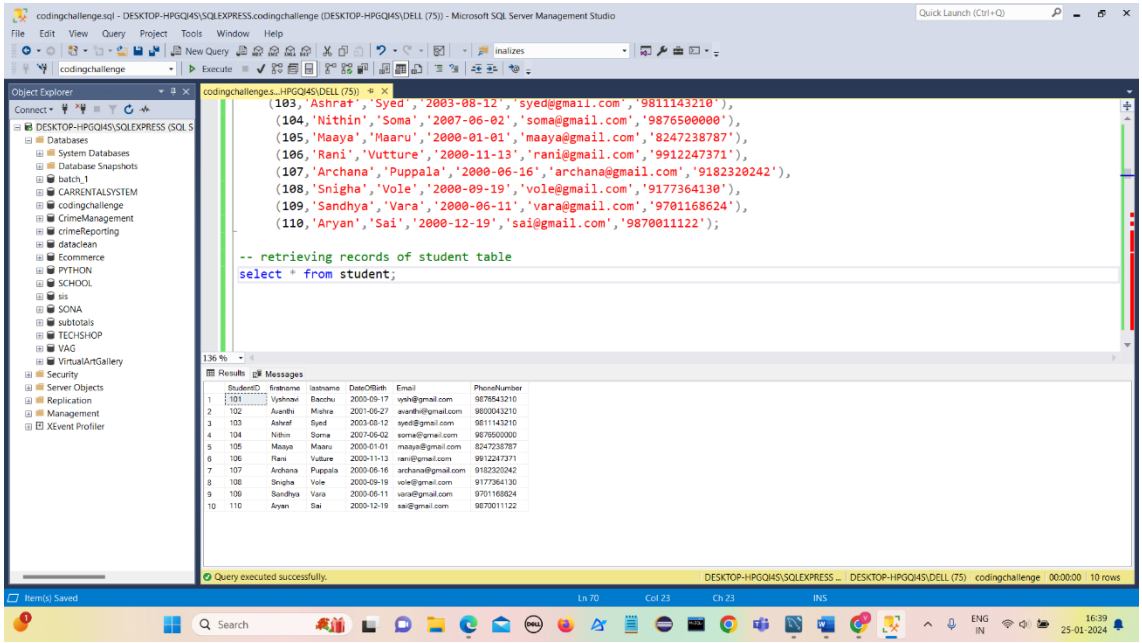


6) Insert records into Student table using insert statement

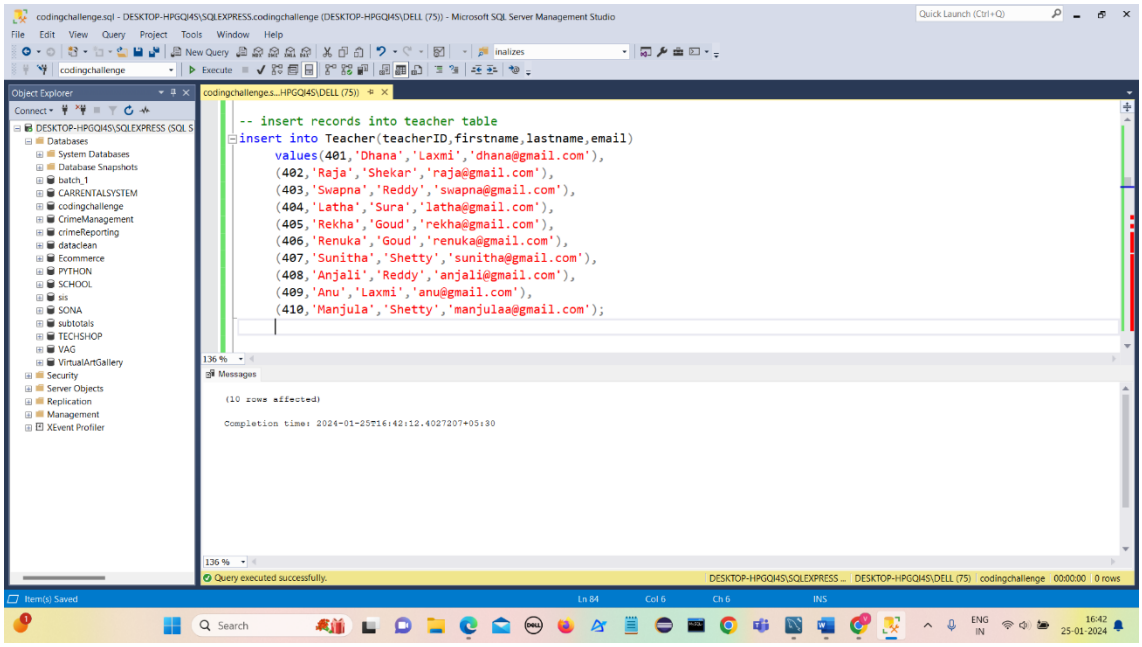


Retrieve records from Student table using select statement

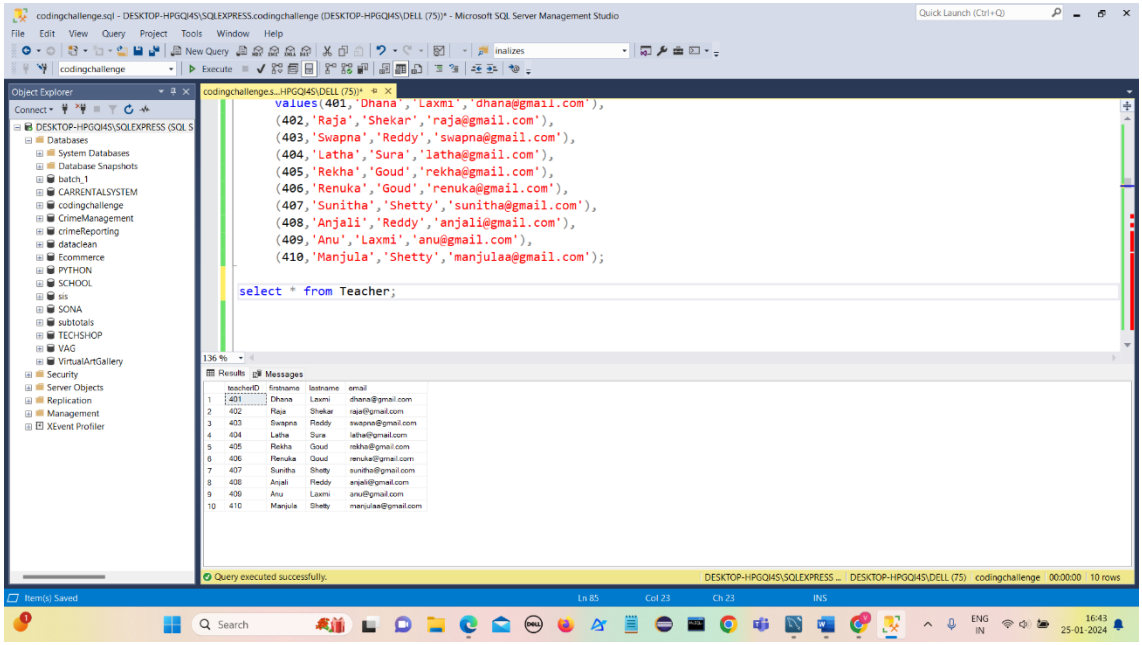
select * from Student;



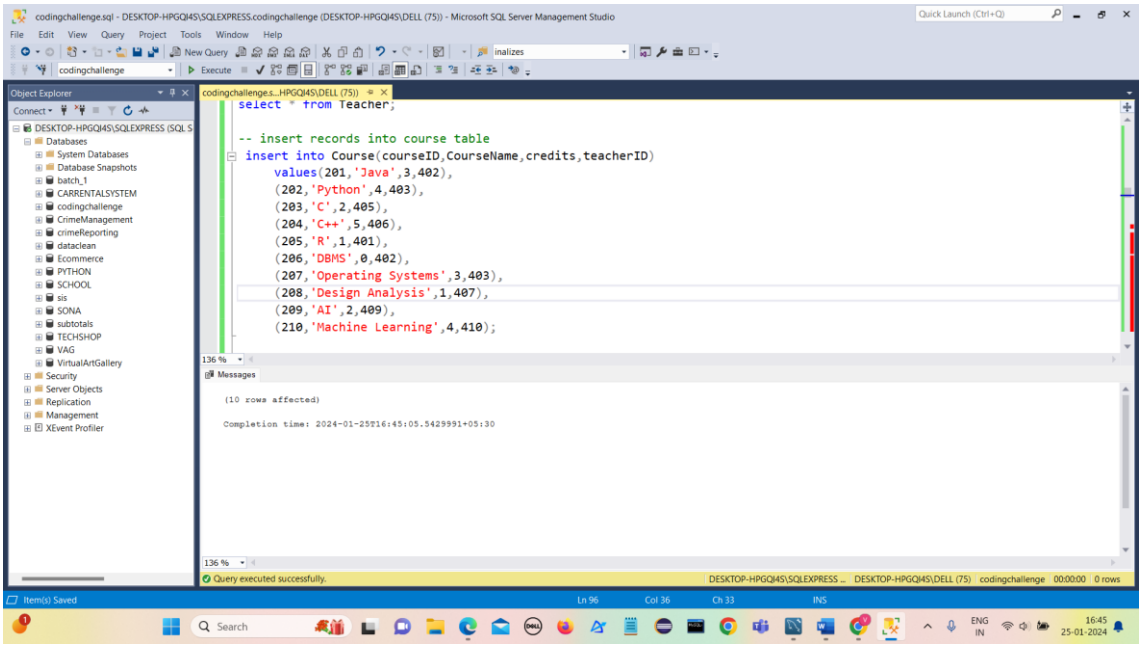
7) Insert records into teacher table



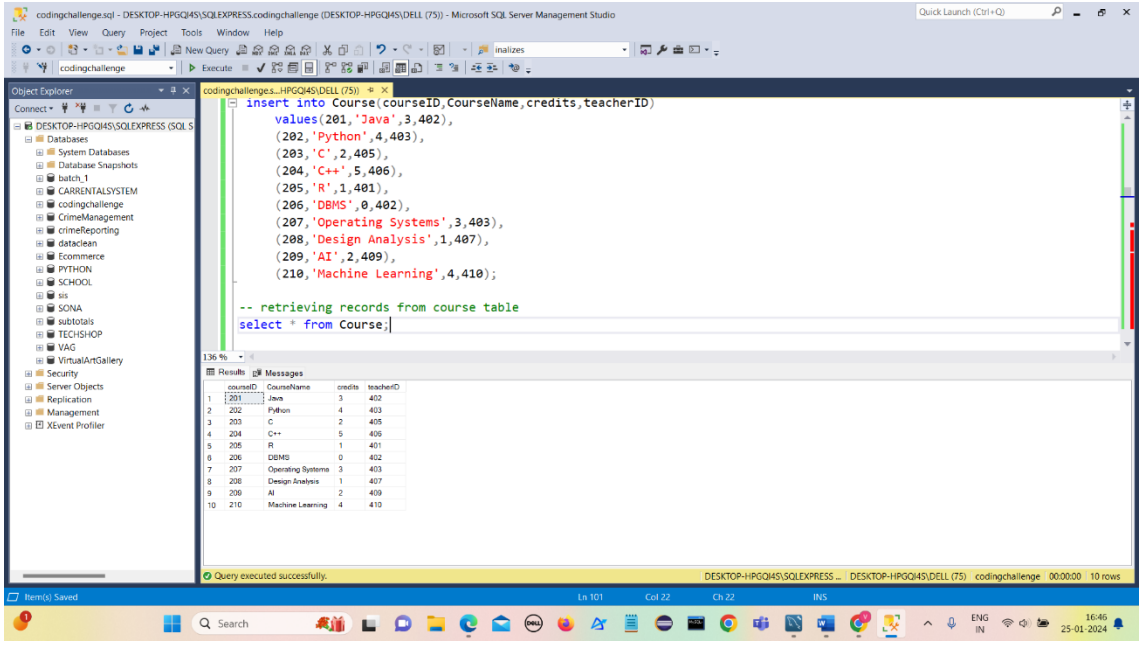
Retrieving records from teacher table



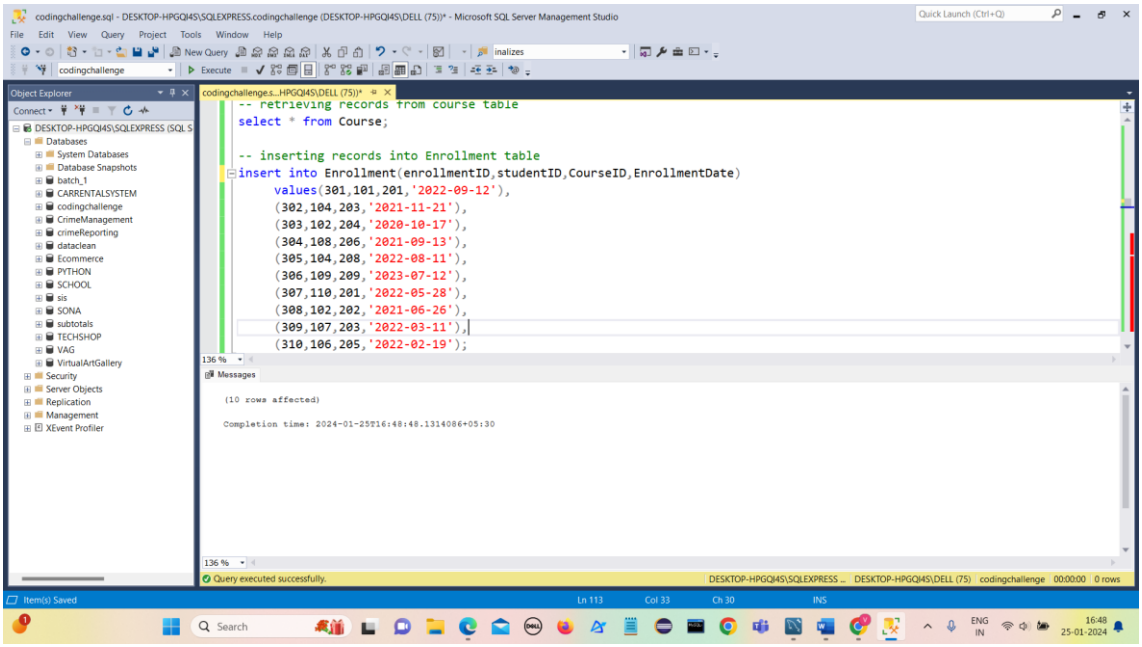
8) Insert records into Course table



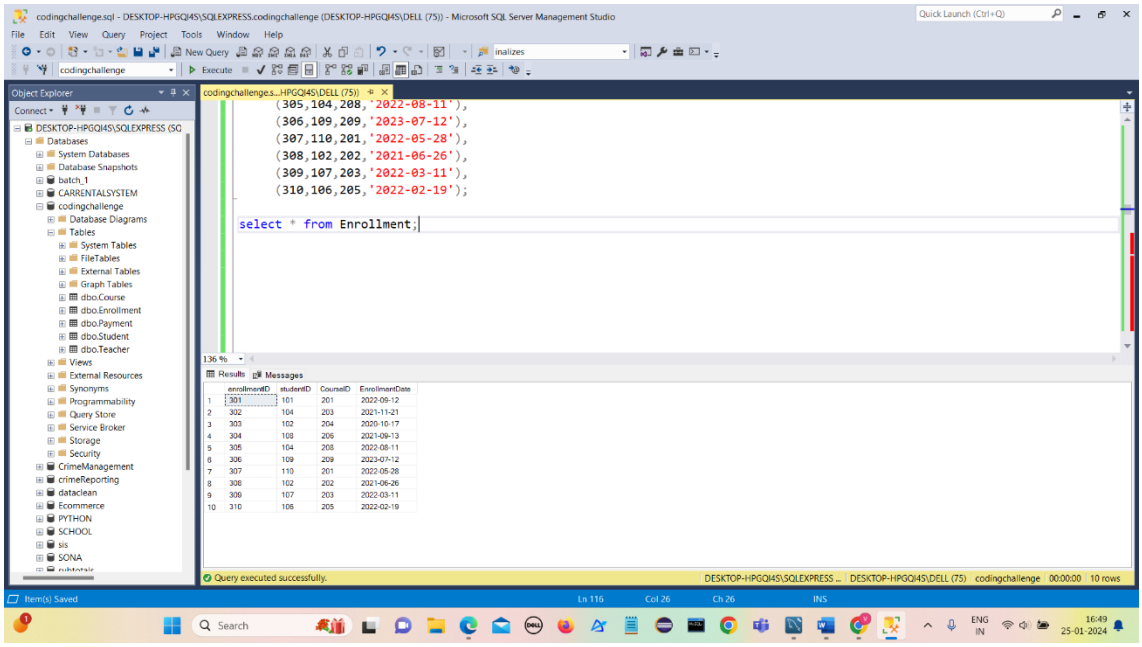
Retrieving records from Course table



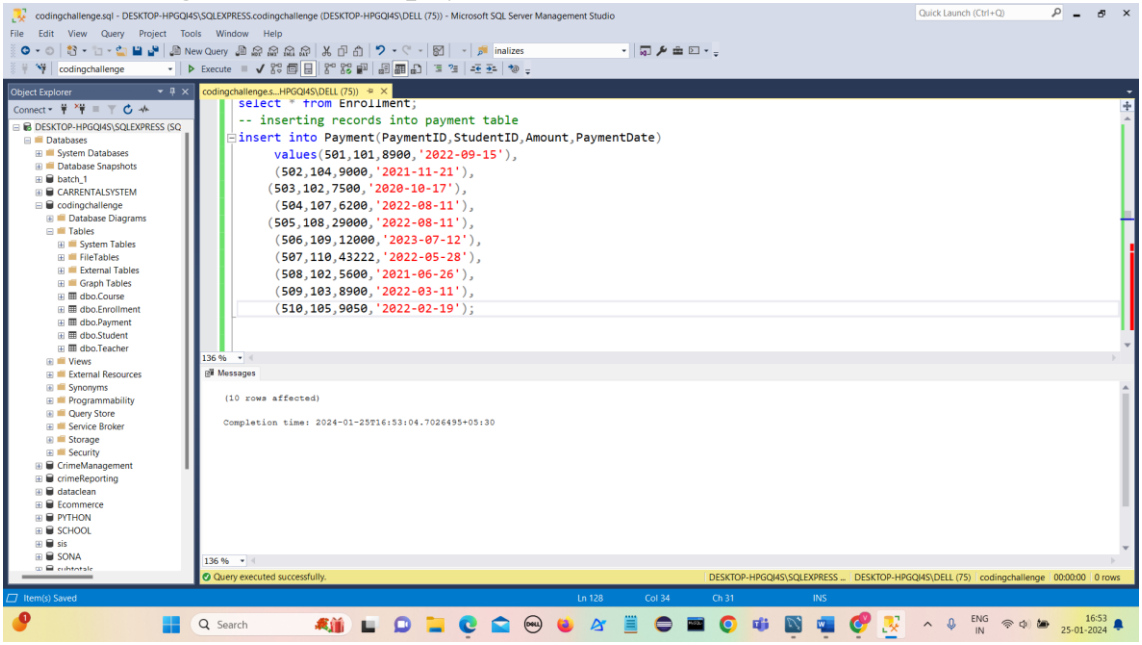
9) Inserting records into Enrollment table



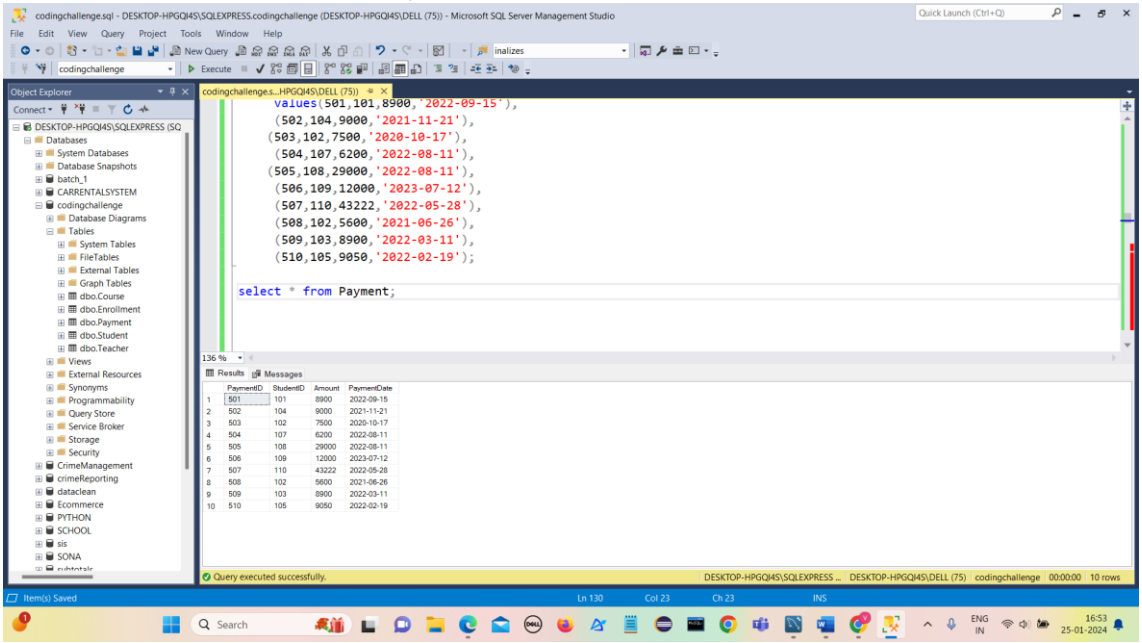
Retrieving records from Enrollment Table



10) Inserting records into payment table



Retrieve records from Payment table



A) Execute OVER and PARTITION BY Clause in SQL Queries

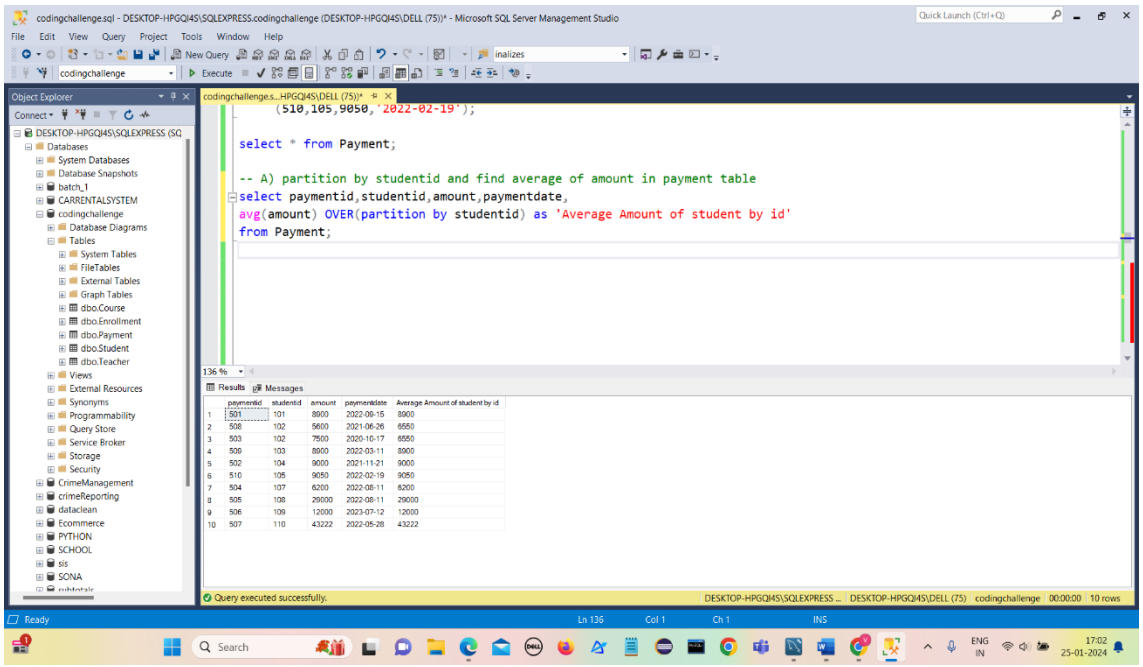
- The 'PARTITION BY' clause in SQL is a subclause of the 'OVER' clause.
- It is used to split a large table into smaller, more manageable partitions to perform calculations easily
- Each partition is then processed for the function present in the 'OVER()' clause.
- Partition is done based on the column name by which we want to divide into groups.

Syntax:

Select col_name,aggregate_function(col_name)
over(partition by col_name)
from table_name;

Example:

partition by studentid and find average of amount in payment table



This query partition by studentid to find average of amount in payments gives paymentid,studentid,amount,paymentdate and average amount obtained by partitioning or dividing into groups by studentid.

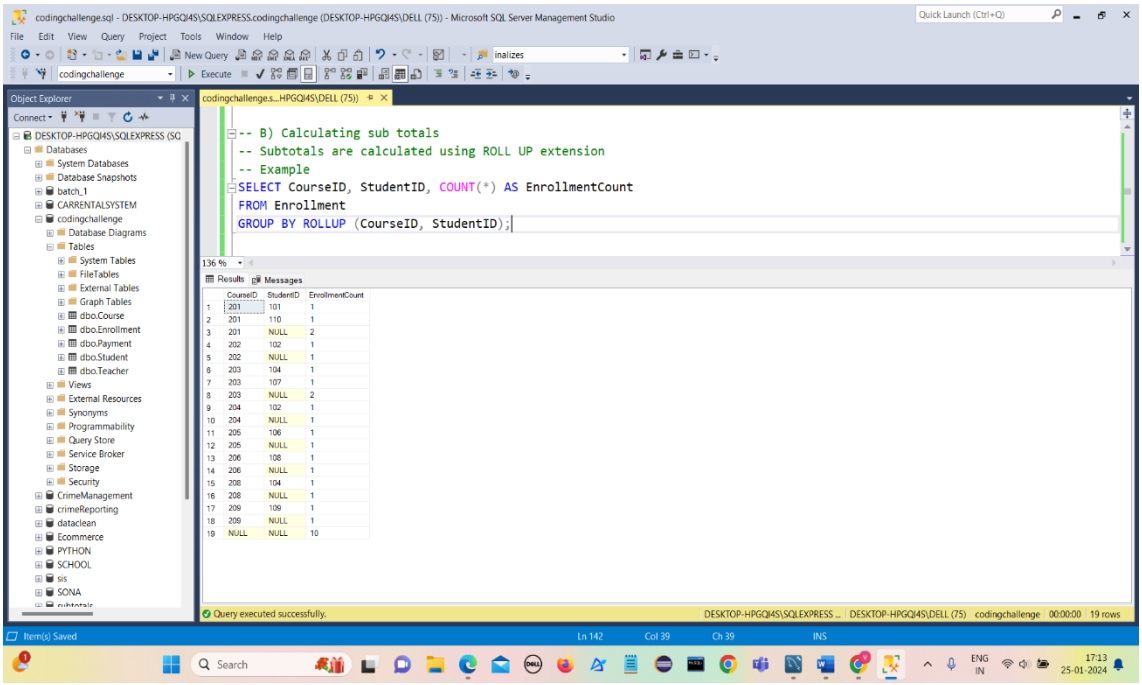
In this query partition is done by studentid.

B) Creating Sub Totals

- Sub total is the sum of similar sets of data but it does not indicate the final total.
- Calculating a subtotal in SQL query can be a bit complicated than the common aggregate queries
- Sub totals can be calculated using a roll up group by extension.

- The **ROLLUP** extension allows us to generate hierarchical subtotal rows according to its input columns and it also adds a grand total row to the result set.

Example:



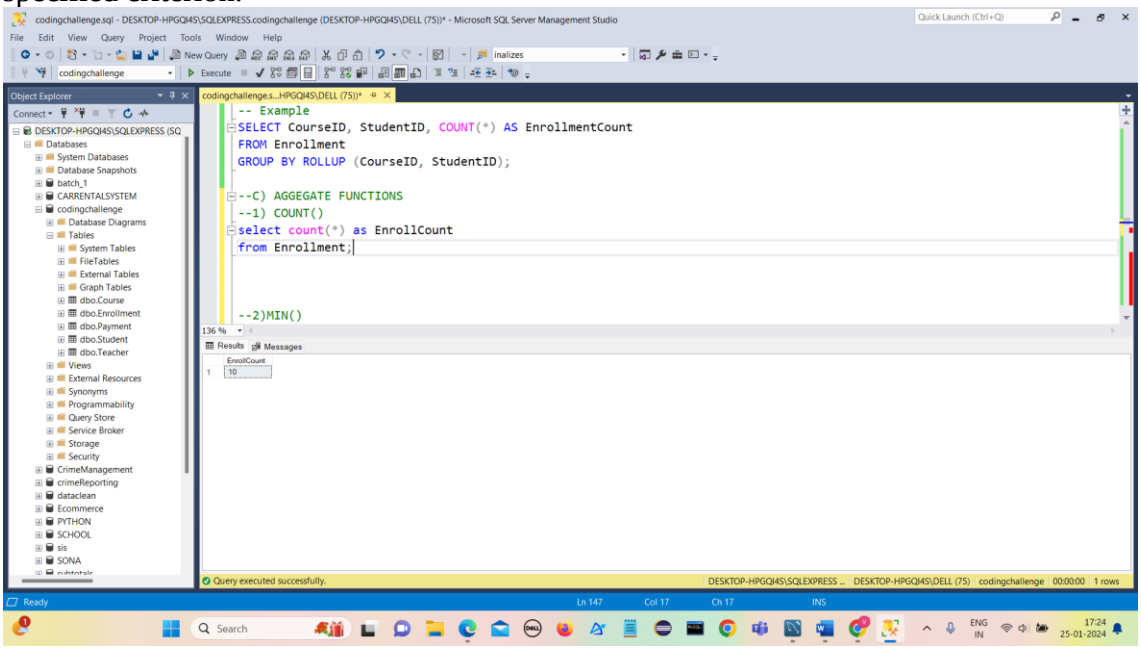
In this query to calculate sub total group by rollup is used where group by is done using CourseID and StudentID which gives the subtotal and total count of students enrolled in courses.All the subtotals are summed together to give grand total.

C) Total Aggregations in SQL

An SQL aggregate function calculates operations on a column of values and returns a single value.
Aggregate functions include:

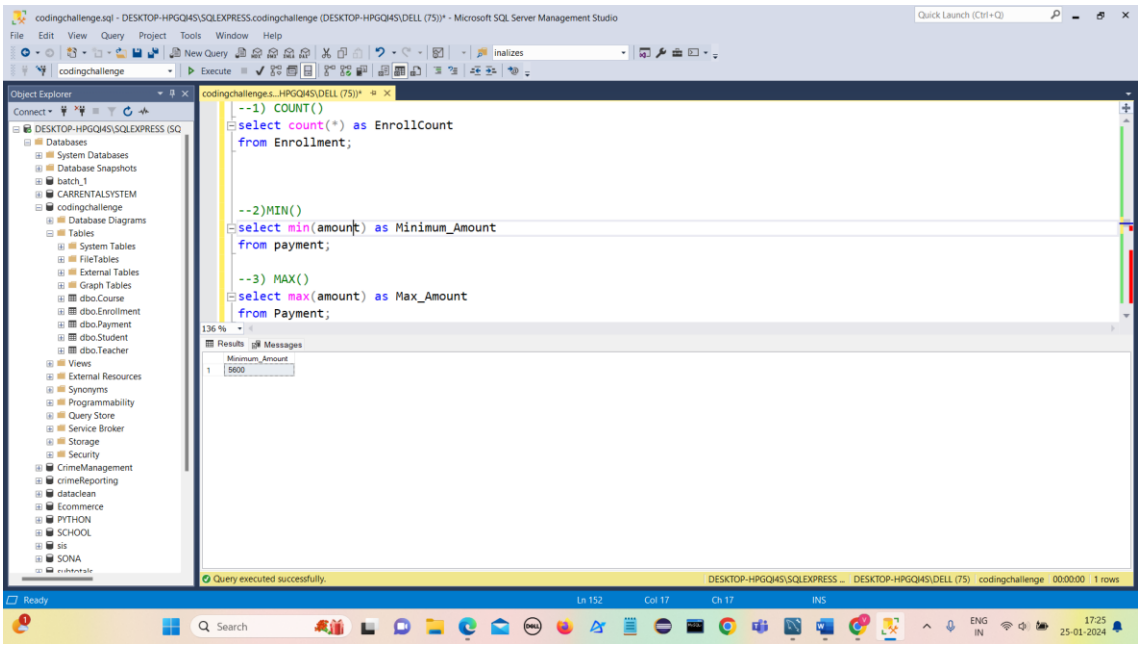
1) **COUNT()**
2) **SUM()**
3) **MIN()**
4) **MAX()**
5) **AVG()**

1) **COUNT()-**The COUNT() function returns the number of rows that matches a specified criterion.



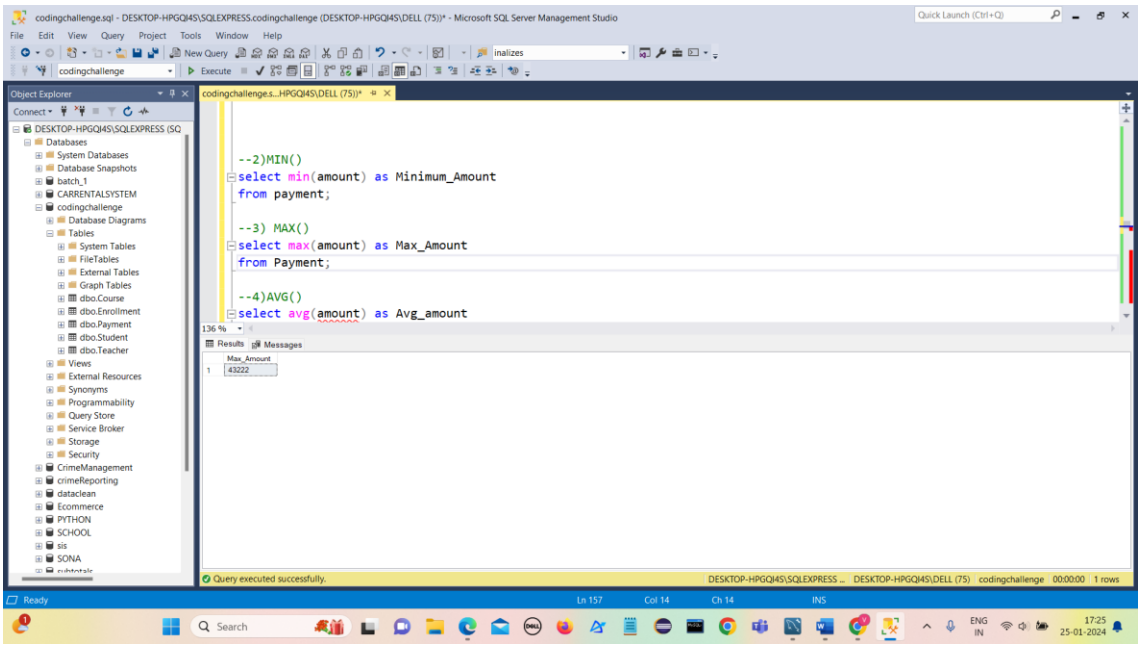
This query gives the total records in enrolment table

2) **MIN()-**The MIN() function returns the smallest value of the selected column.



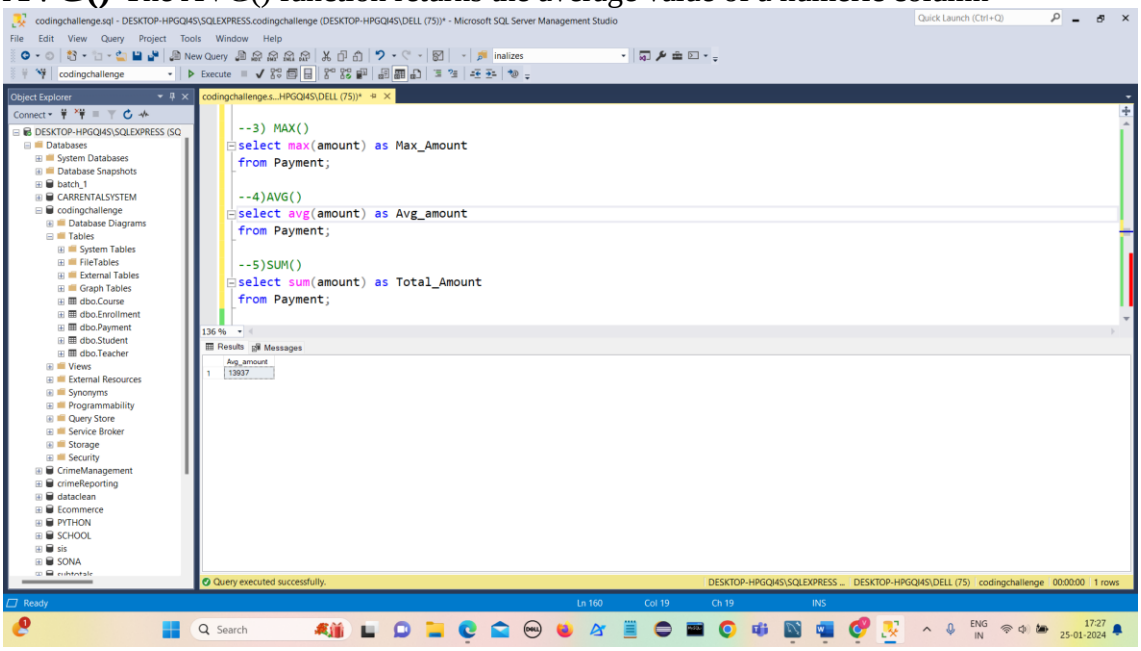
This query gives the minimum of salary of payments

3) **MAX()-**The MAX() function returns the largest value of the selected column.



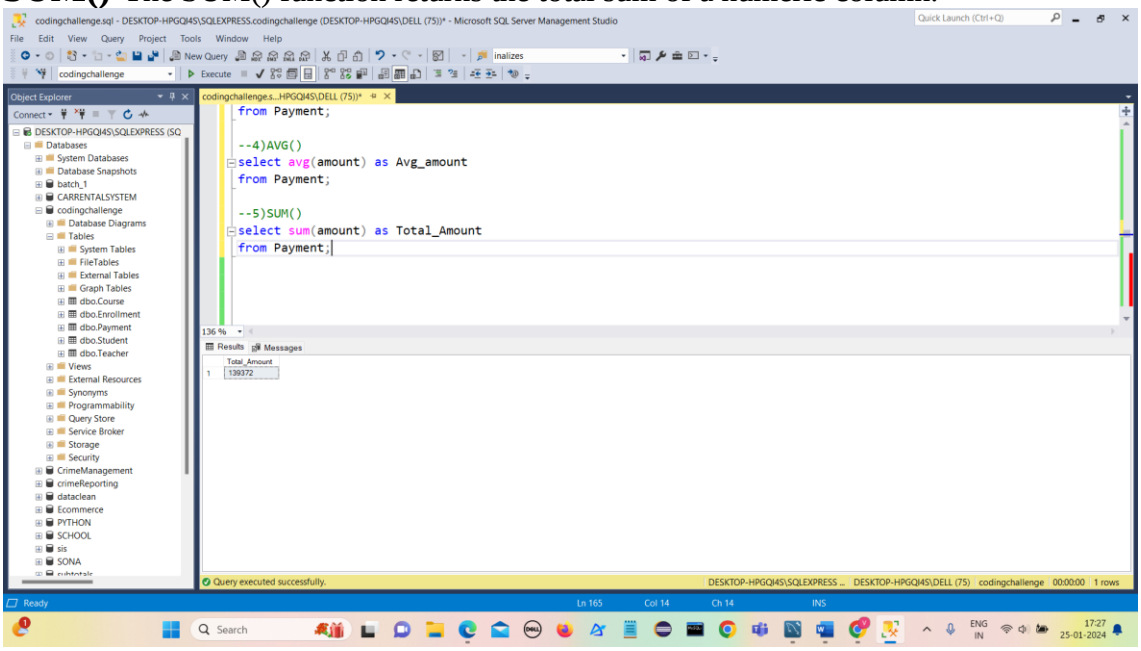
This query gives the maximum of salary of payments

4) **AVG()-**The AVG() function returns the average value of a numeric column



This query gives the average of salary of payments

5) **SUM()-**The SUM() function returns the total sum of a numeric column.



This query gives the sum of salary of payments