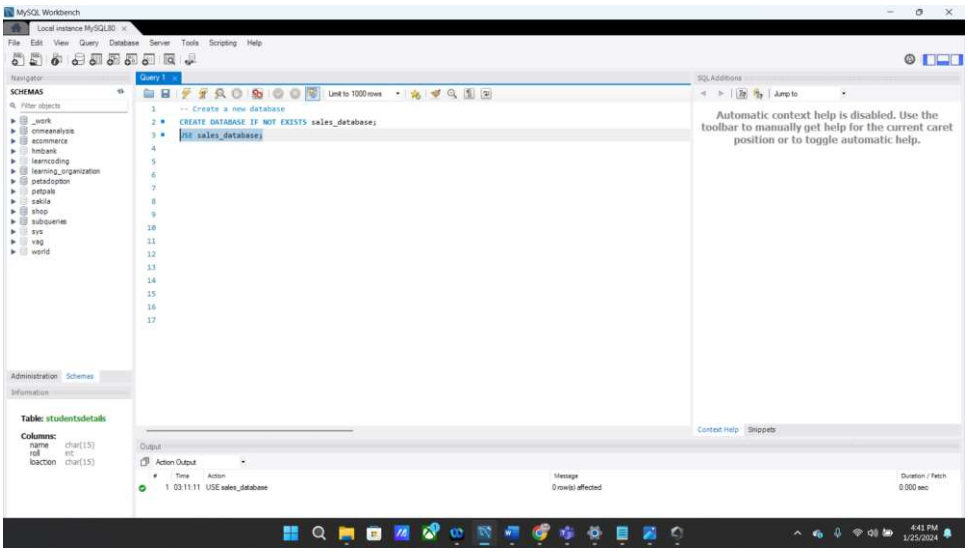
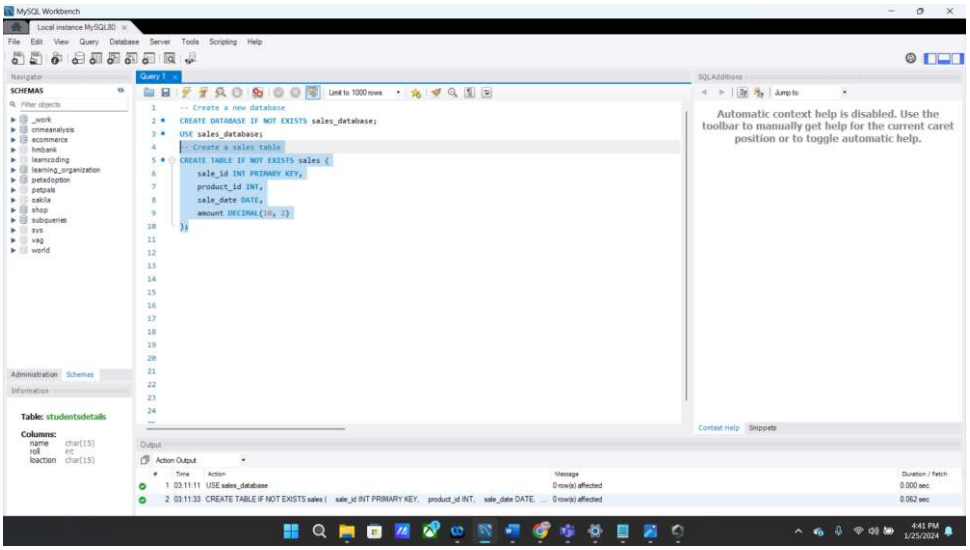
NAME-Jayant Jhunja

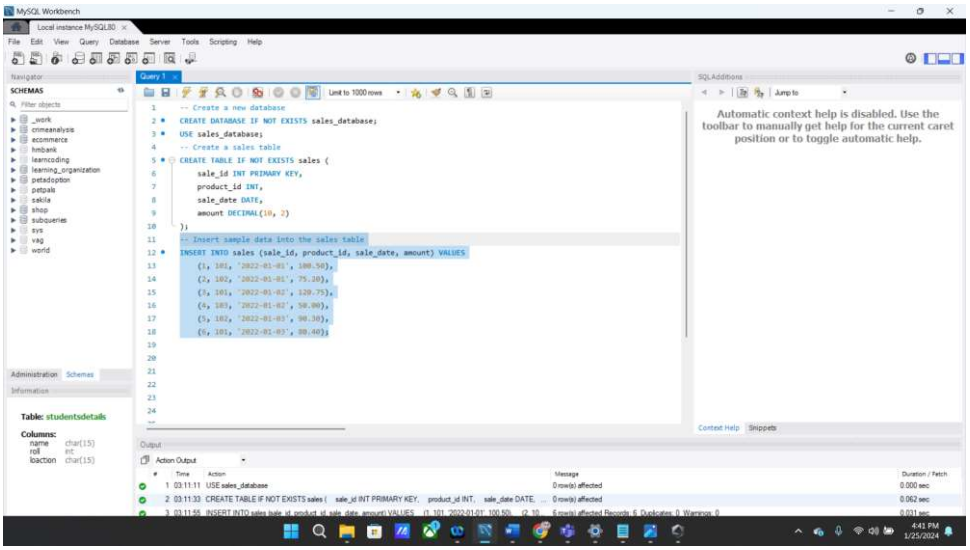
SQL

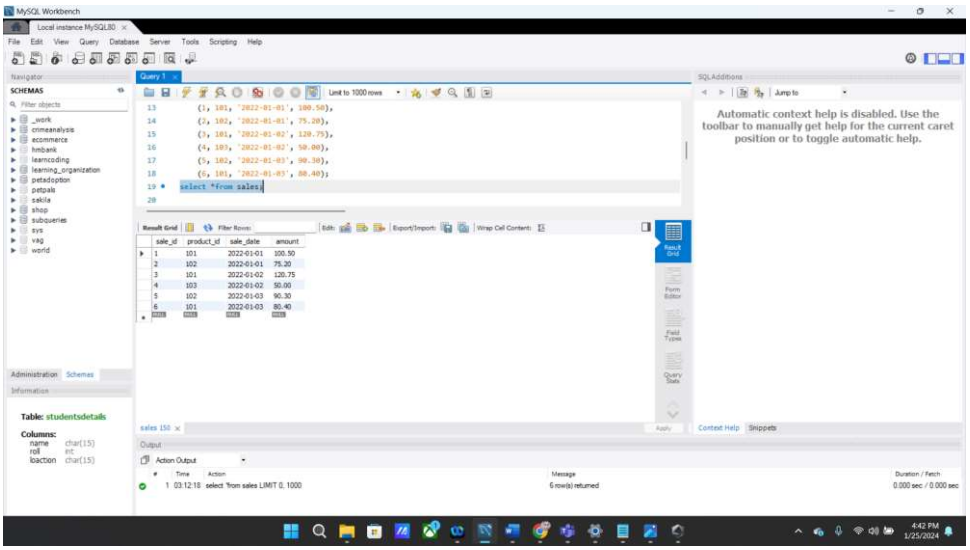
CODING CHALLENGE

The OVER and PARTITION BY clauses in SQL are often used with window functions to perform calculations across a subset of rows.




MySQL Workbench
-- Create a new database
CREATE DATABASE IF NOT EXISTS sales_database;
USE sales_database;
-- Create a sales table
CREATE TABLE IF NOT EXISTS sales (
sale_id INT PRIMARY KEY,
product_id INT,
sale_date DATE,
amount DECIMAL(10, 2)
);
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

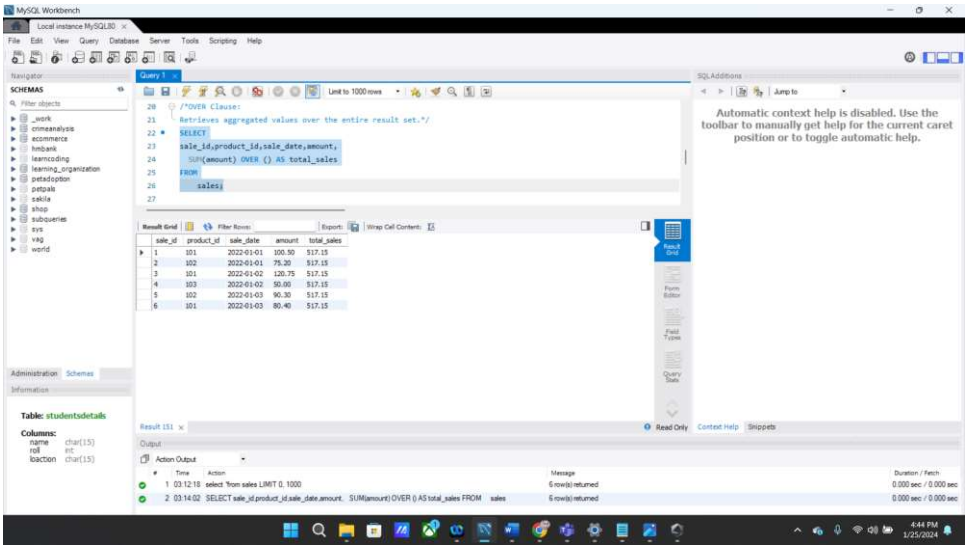

MySQL Workbench
-- Create a new database
CREATE DATABASE IF NOT EXISTS sales_database;
USE sales_database;
-- Create a sales table
CREATE TABLE IF NOT EXISTS sales (
sale_id INT PRIMARY KEY,
product_id INT,
sale_date DATE,
amount DECIMAL(10, 2)
);
-- Insert sample data into the sales table
INSERT INTO sales (sale_id, product_id, sale_date, amount) VALUES
(1, 101, '2022-01-01', 100.50),
(2, 102, '2022-01-01', 75.20),
(3, 101, '2022-01-02', 120.75),
(4, 103, '2022-01-02', 50.00),
(5, 102, '2022-01-03', 90.30),
(6, 101, '2022-01-03', 80.40);
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.


MySQL Workbench
(1, 101, '2022-01-01', 100.50),
(2, 102, '2022-01-01', 75.20),
(3, 101, '2022-01-02', 120.75),
(4, 103, '2022-01-02', 50.00),
(5, 102, '2022-01-03', 90.30),
(6, 101, '2022-01-03', 80.40);
select *from sales;
sale_id product_id sale_date amount
1 101 2022-01-01 100.50
2 102 2022-01-01 75.20
3 101 2022-01-02 120.75
4 103 2022-01-02 50.00
5 102 2022-01-03 90.30
6 101 2022-01-03 80.40
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

OVER Clause:

Retrieves aggregated values over the entire result set.

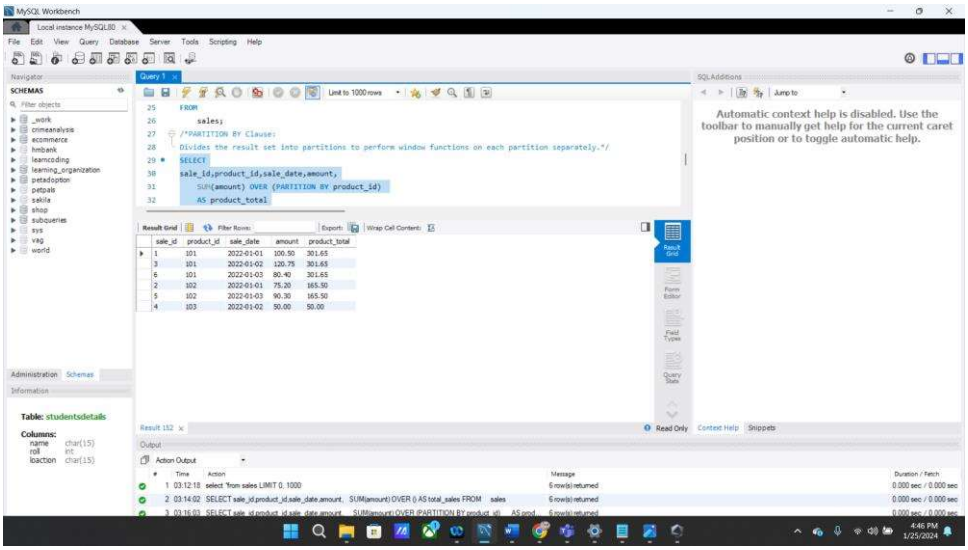
1.



PARTITION BY Clause:

Divides the result set into partitions to perform window functions on each partition separately.

2.

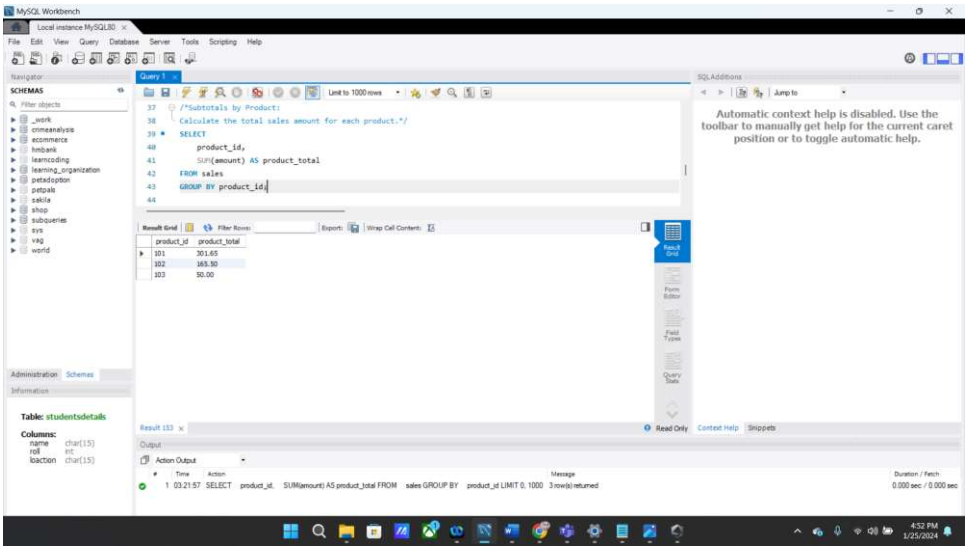


In the first query, the OVER clause without PARTITION BY calculates the total sales amount over the entire result set for each row. In the second query, the PARTITION BY clause is used to calculate the total sales amount for each product separately.

* To create subtotals and total aggregations in MYSQL, we can use the GROUP BY clause along with aggregate functions.

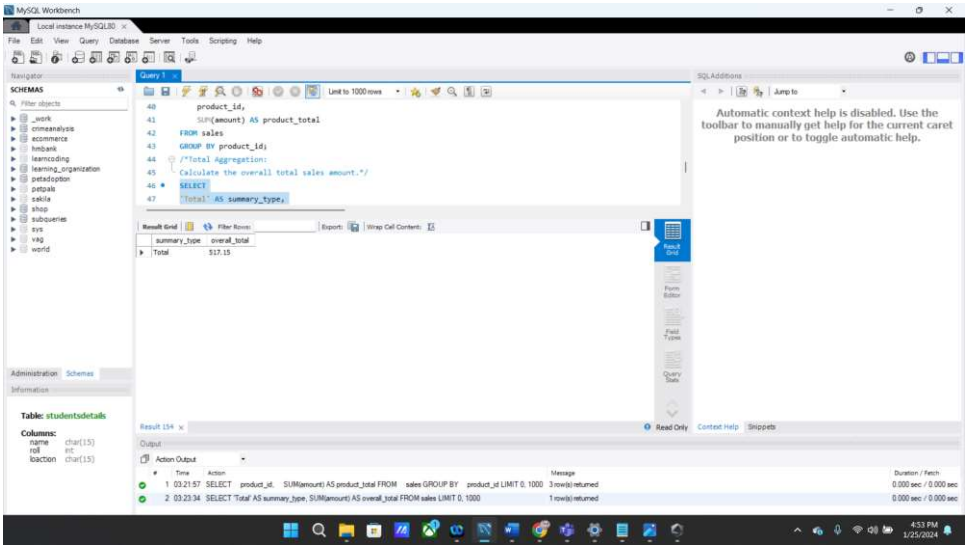
-Subtotals by Product:

Calculate the total sales amount for each product.



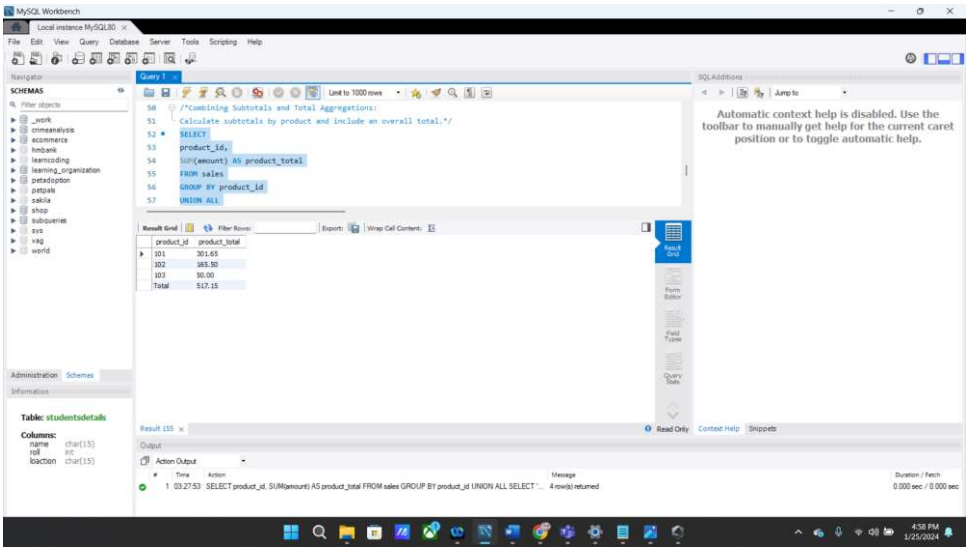
--Total Aggregation:

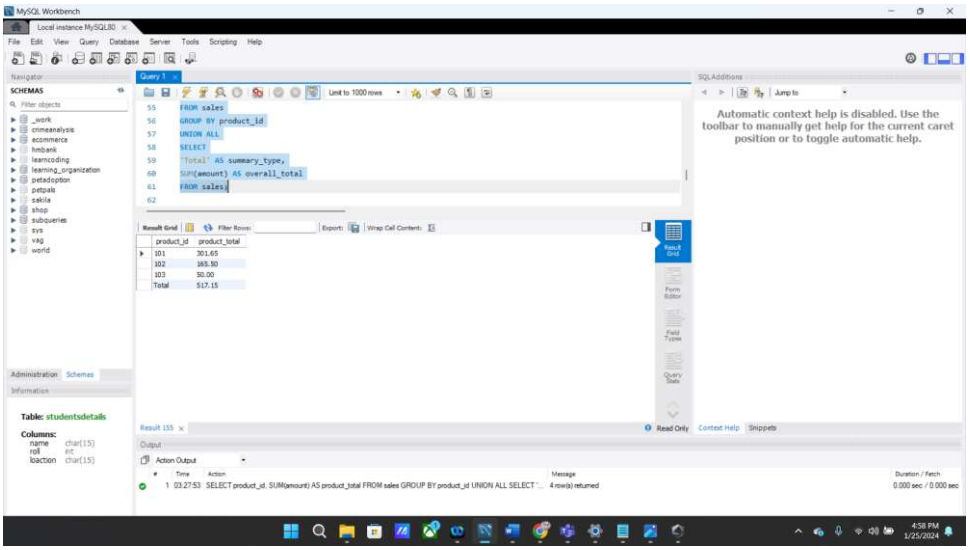
Calculate the overall total sales amount.



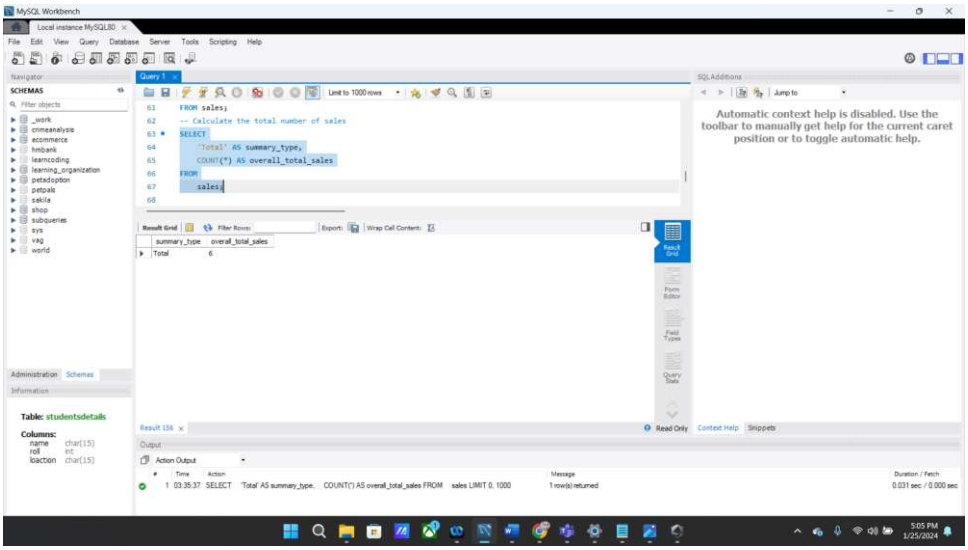
--Combining Subtotals and Total Aggregations:

Calculate subtotals by product and include an overall total.

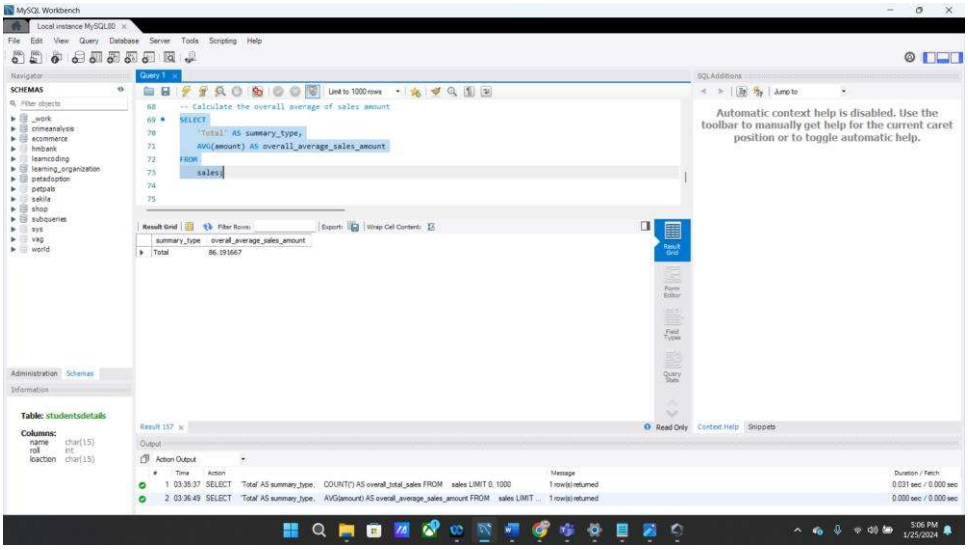




* If we want to retrieve the total count of rows in the sales table, we can use the COUNT aggregate function

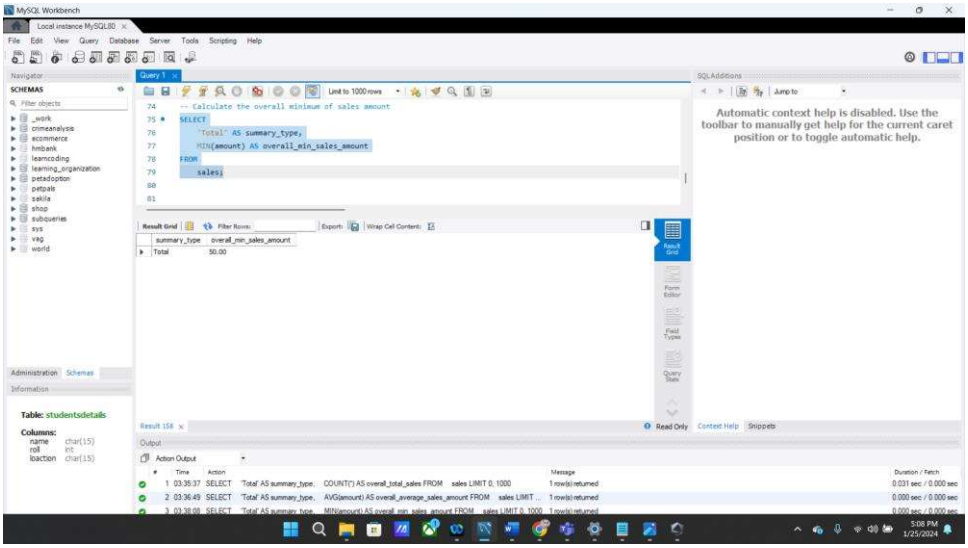


*If we want to retrieve the average of a specific column, such as the amount column, in the sales table, we can use the AVG aggregate function.



*If we want to retrieve the minimum value of a specific column, such as the amount

column, in the sales table, we can use the MIN aggregate function.



*If we want to retrieve the maximum value of a specific column, such as the amount column, in the sales table, we can use the MAX aggregate function.