**Q1. SELECT clause with WHERE, AND, DISTINCT, Wild Card (LIKE)**

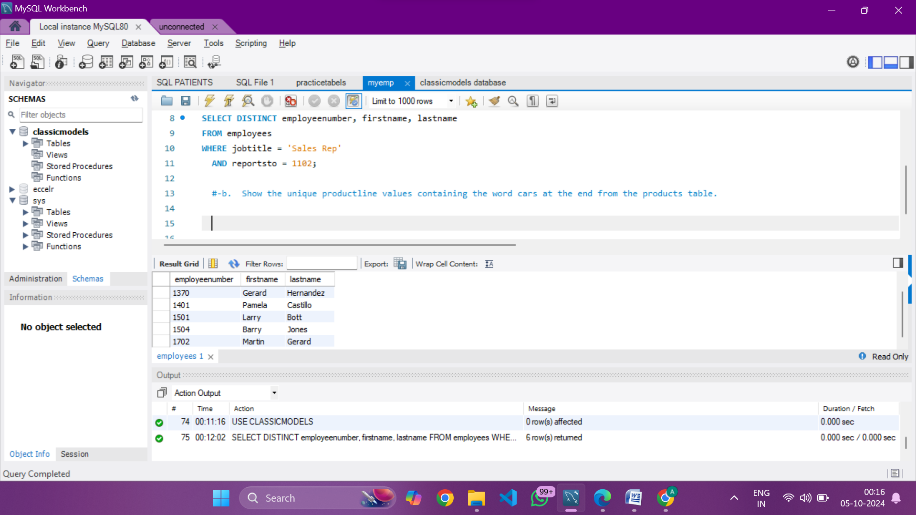
**A – Answer**

**SELECT DISTINCT employeenumber, firstname, lastname**

**FROM employees**

**WHERE jobtitle = 'Sales Rep'**

**AND reportsto = 1102;**

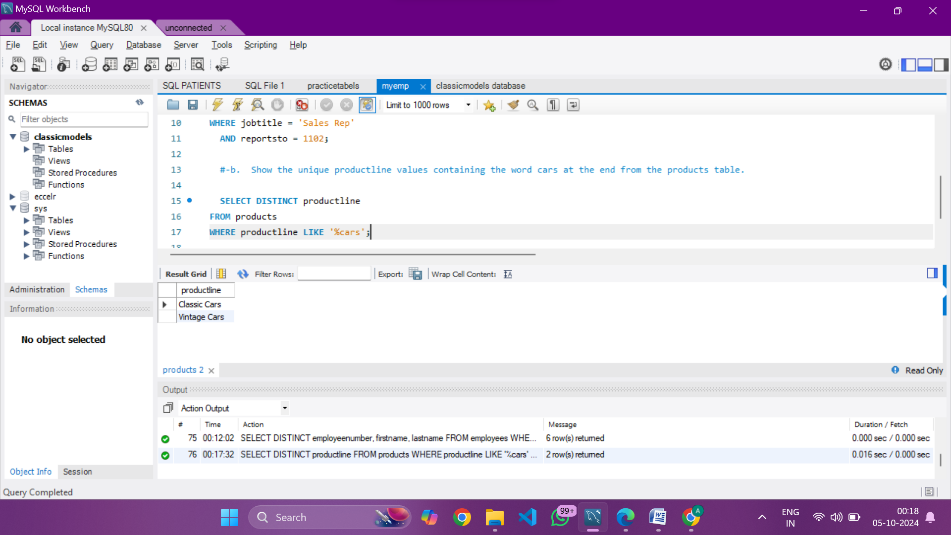
****

B -Answer

SELECT DISTINCT productline

FROM products

WHERE productline LIKE '%cars';



**Q2. Group By with Aggregation functions and Having clause, Date and Time functions**

**a-answer**

SELECT productCode,

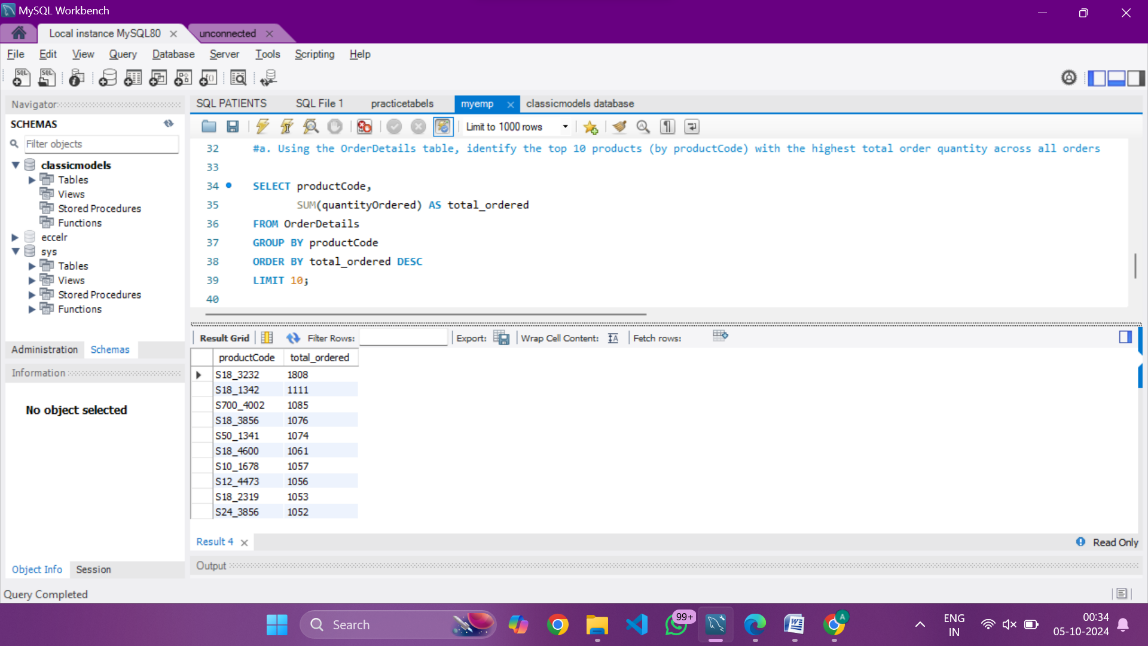
SUM(quantityOrdered) AS total\_ordered

FROM OrderDetails

GROUP BY productCode

ORDER BY total\_ordered DESC

LIMIT 10;



**Q3**

b-answer

SELECT MONTHNAME(PAYMENTDATE) AS MONTH,

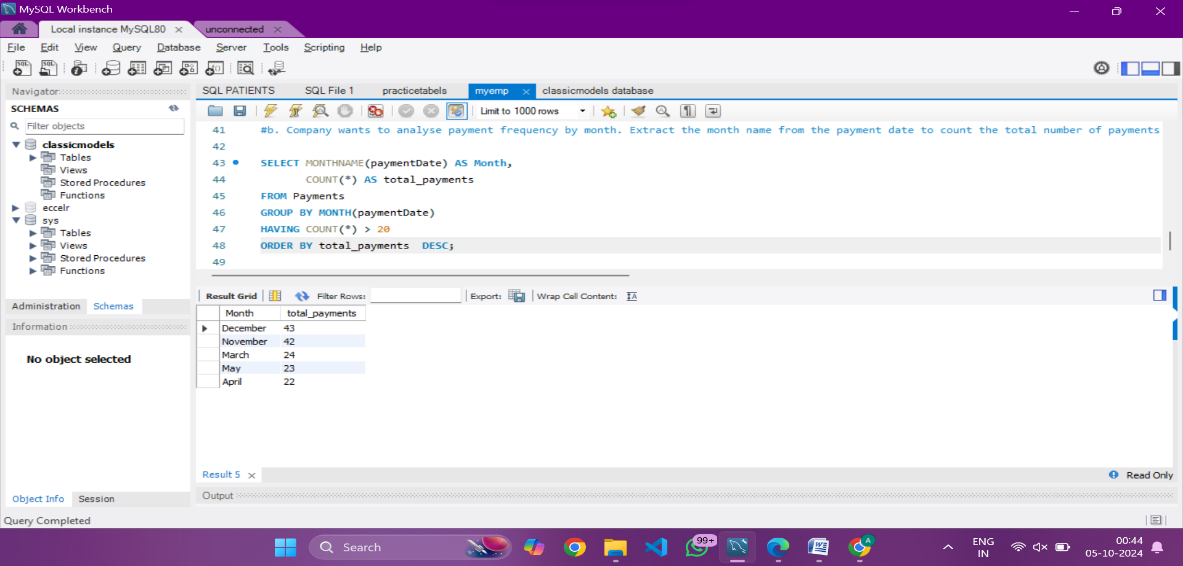
COUNT(\*) AS total\_payments

FROM PAYMENTS

GROUP BY MONTH(PAYMENTDATE)

HAVING COUNT(\*)>20

ORDER BY total\_payments DESC;



**Q4. CONSTRAINTS: Primary, key, foreign key, Unique, check, not null, default**

**a-answer**

USE Customers\_Orders;

CREATE TABLE Customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(255) UNIQUE,

phone\_number VARCHAR(20)

);

SELECT \* FROM Customers;

**Q4**

**b-answer**

CREATE TABLE Orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

total\_amount DECIMAL(10, 2),

CONSTRAINT fk\_customer

FOREIGN KEY (customer\_id)

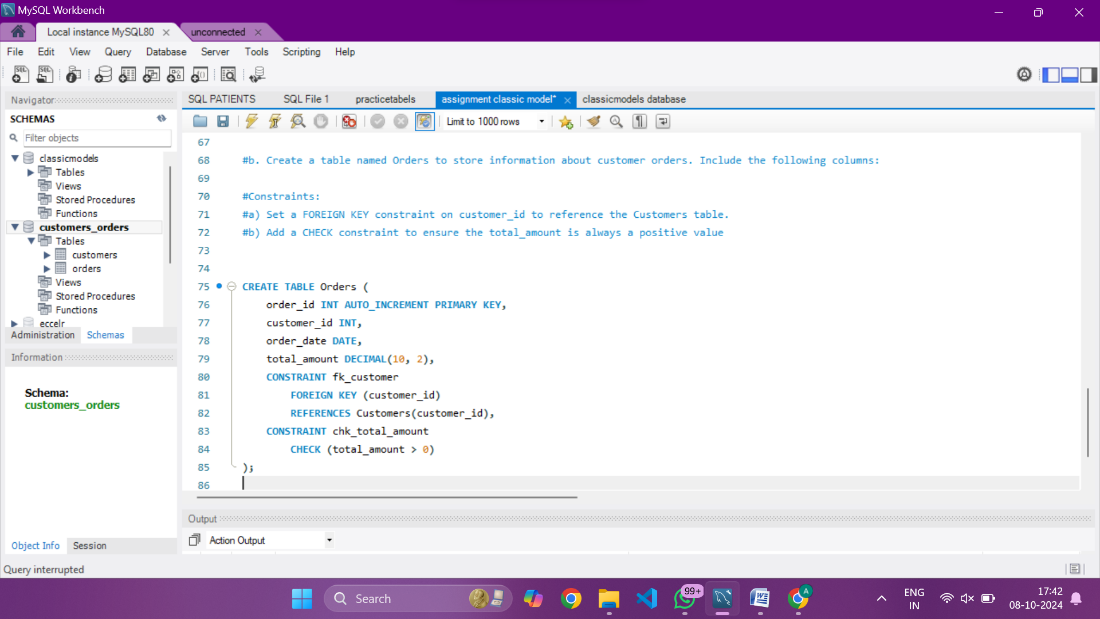
REFERENCES Customers(customer\_id),

CONSTRAINT chk\_total\_amount

CHECK (total\_amount > 0)

);

select \* from orders;



**Q5. JOINS**

**a-answer**

use classicmodels;

SELECT C.country, COUNT(O.orderNumber) AS order\_count

FROM Customers C

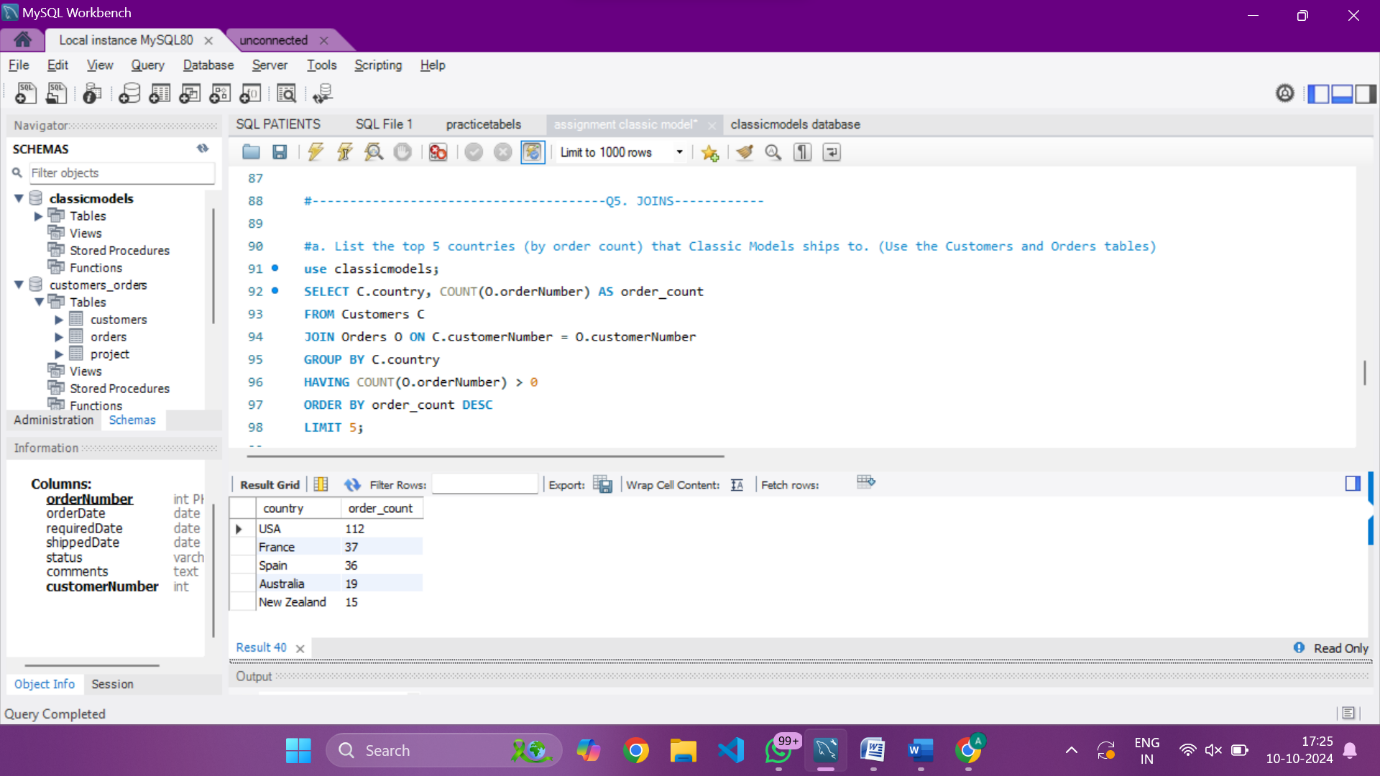
JOIN Orders O ON C.customerNumber = O.customerNumber

GROUP BY C.country

HAVING COUNT(O.orderNumber) > 0

ORDER BY order\_count DESC

LIMIT 5;

****

**Q6. SELF JOIN**

**a-answer**

use customer\_order;

CREATE TABLE project (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

FullName VARCHAR(50) NOT NULL,

Gender ENUM('Male', 'Female') NOT NULL,

ManagerID INT

);

#---adding data

INSERT INTO project (FullName, Gender, ManagerID)

VALUES

('Pranaya', 'Male', 3),

('Priyanka', 'Female', 1),

('Preety', 'Female', NULL),

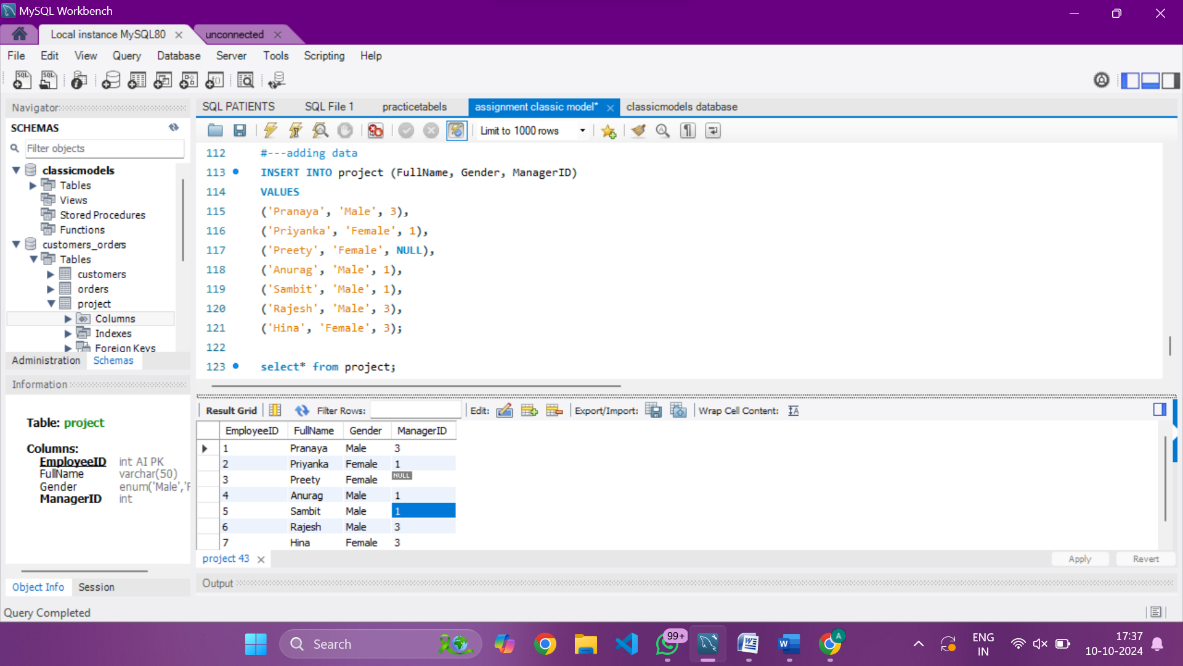
('Anurag', 'Male', 1),

('Sambit', 'Male', 1),

('Rajesh', 'Male', 3),

('Hina', 'Female', 3);

select\* from project;



update project set ManagerID = 3

where EmployeeID = 1; -updating the pranay’s manager id

#expected output----

SELECT

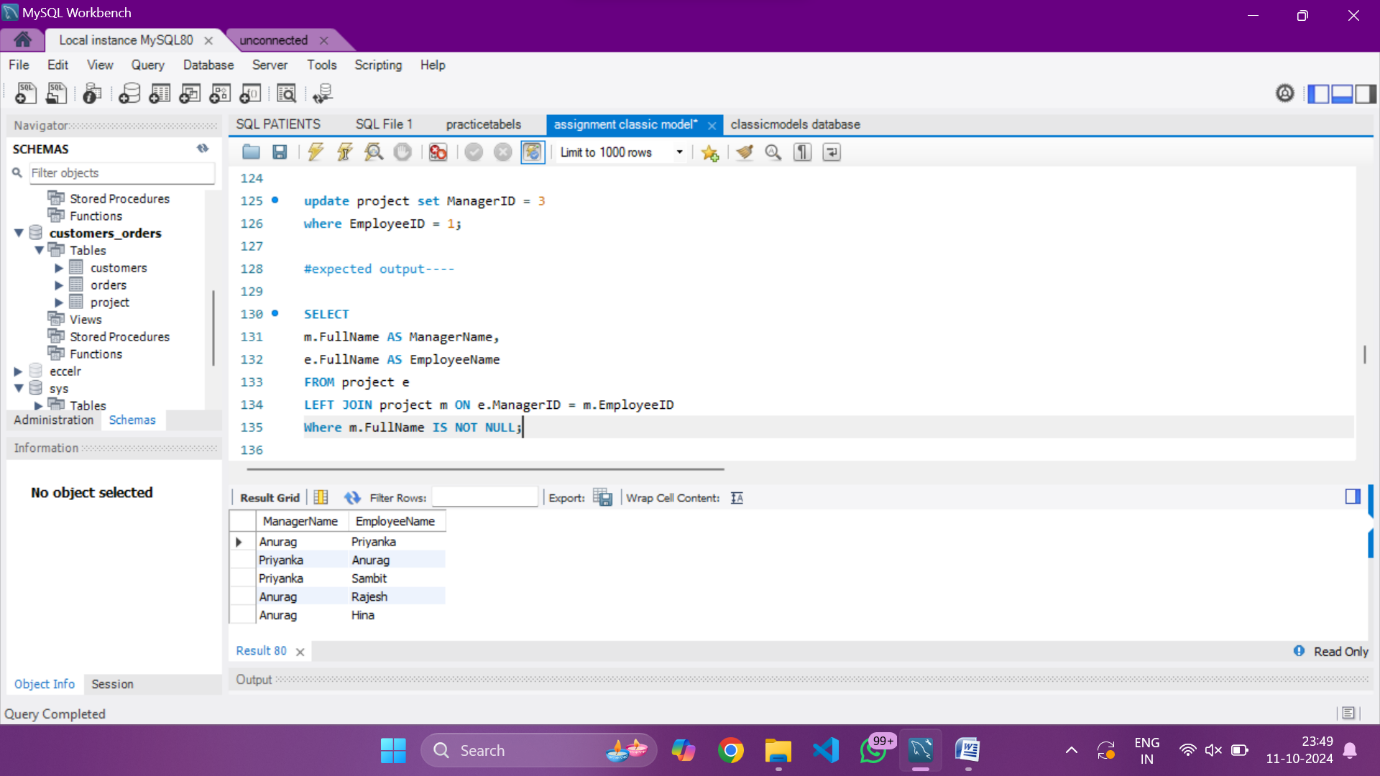
m.FullName AS ManagerName,

e.FullName AS EmployeeName

FROM project e

LEFT JOIN project m ON e.ManagerID = m.EmployeeID

Where m.FullName IS NOT NULL;



**Q7. DDL Commands: Create, Alter, Rename**

**a-answer**

CREATE TABLE facility (

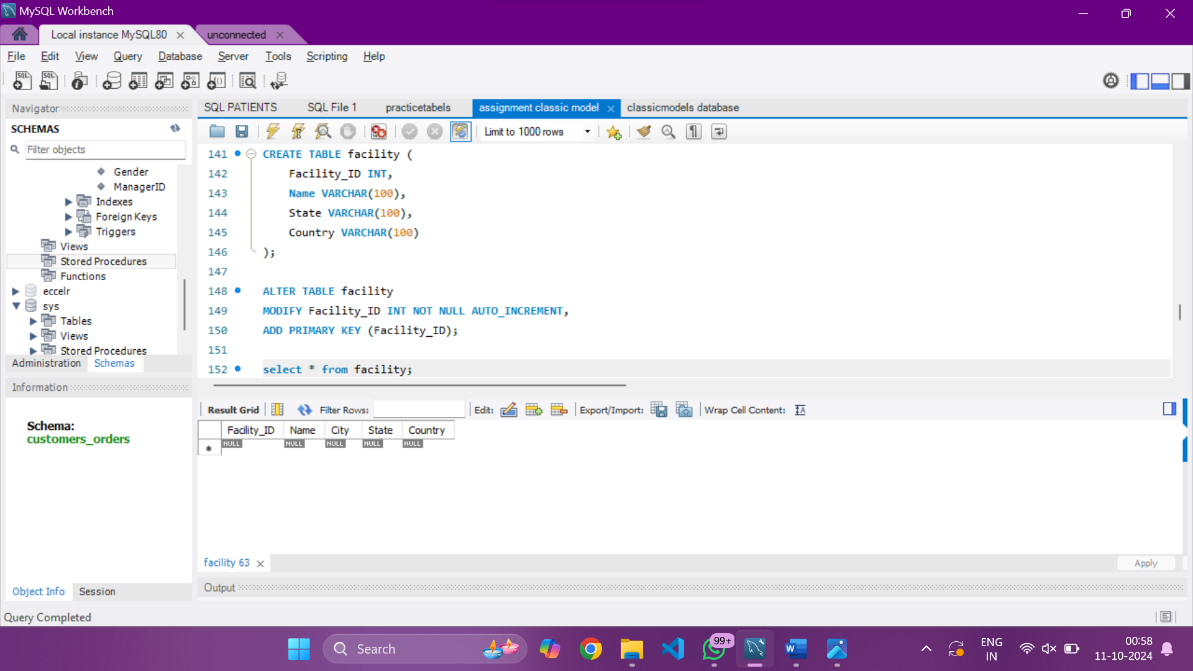
Facility\_ID INT,

Name VARCHAR(100),

State VARCHAR(100),

Country VARCHAR(100)

);



CREATE TABLE facility (

Facility\_ID INT,

Name VARCHAR(100),

State VARCHAR(100),

Country VARCHAR(100)

);

ALTER TABLE facility

MODIFY Facility\_ID INT NOT NULL AUTO\_INCREMENT,

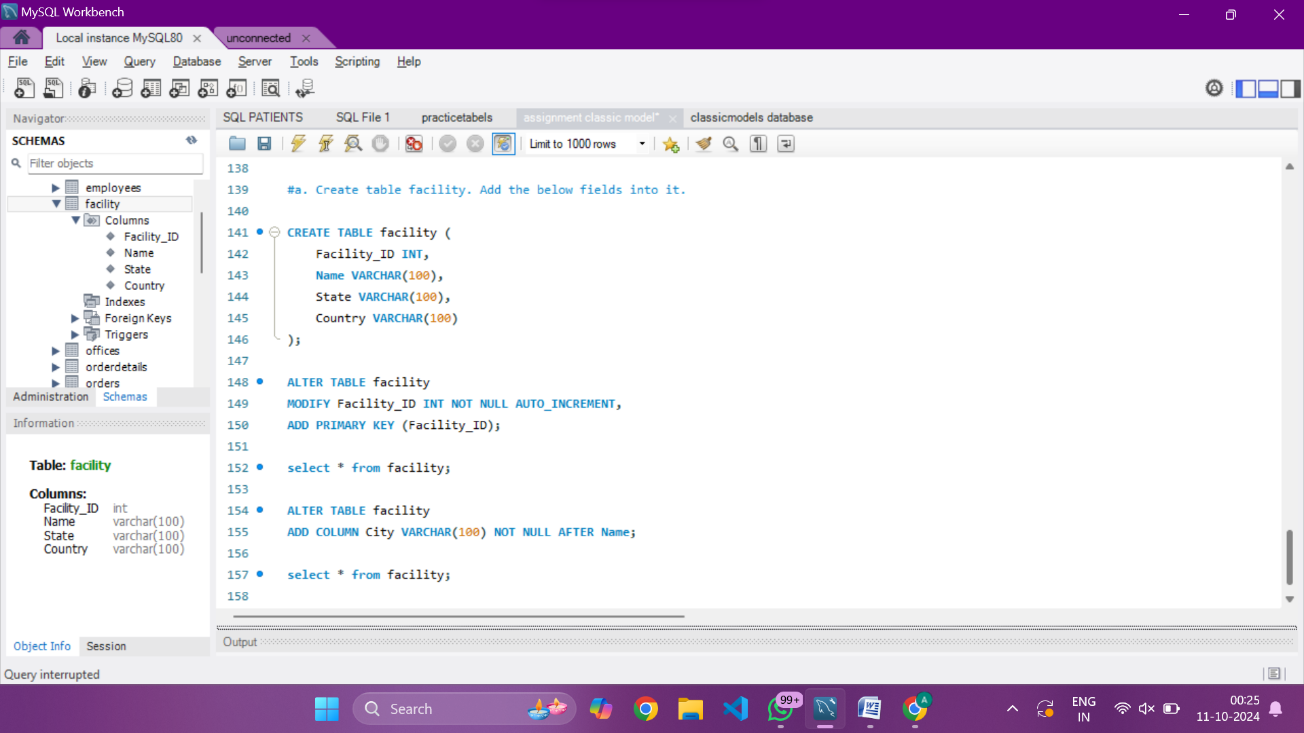
ADD PRIMARY KEY (Facility\_ID);

select \* from facility;

ALTER TABLE facility

ADD COLUMN City VARCHAR(100) NOT NULL AFTER Name;

select \* from facility;



**Q8. Views in SQL**

**-answer**

use classicmodels;

SELECT pl.productLine,

SUM(od.quantityOrdered \* od.priceEach) AS total\_sales,

COUNT(DISTINCT o.orderNumber) AS number\_of\_orders

FROM ProductLines pl

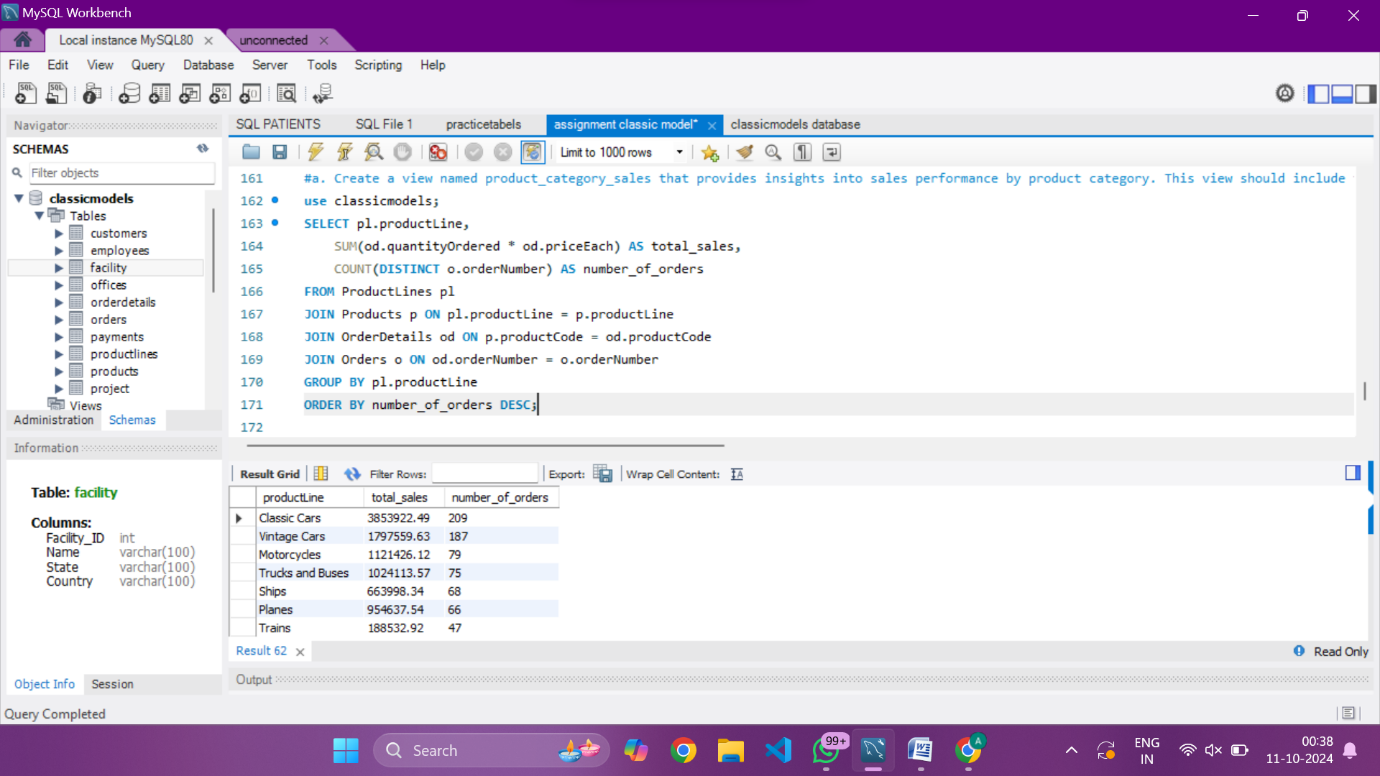
JOIN Products p ON pl.productLine = p.productLine

JOIN OrderDetails od ON p.productCode = od.productCode

JOIN Orders o ON od.orderNumber = o.orderNumber

GROUP BY pl.productLine

ORDER BY number\_of\_orders DESC;



**Q.9. Stored Procedures in SQL with parameters**

a. Create a stored procedure Get\_country\_payments which takes in year and country as inputs and gives year wise, country wise total amount as an output. Format the total amount to nearest thousand unit (K)

**ANSWER-**

DELIMITER //

CREATE PROCEDURE Get\_country\_payments(IN input\_year INT, IN input\_country VARCHAR(255))

BEGIN

SELECT

YEAR(p.PaymentDate) AS Year,

c.Country,

CONCAT(FLOOR(SUM(p.Amount) / 1000), 'K') AS TotalAmount

FROM

Payments p

JOIN

Customers c ON p.CustomerNumber = c.CustomerNumber -- Using CustomerNumber for the join

WHERE

YEAR(p.PaymentDate) = input\_year

AND c.Country = input\_country

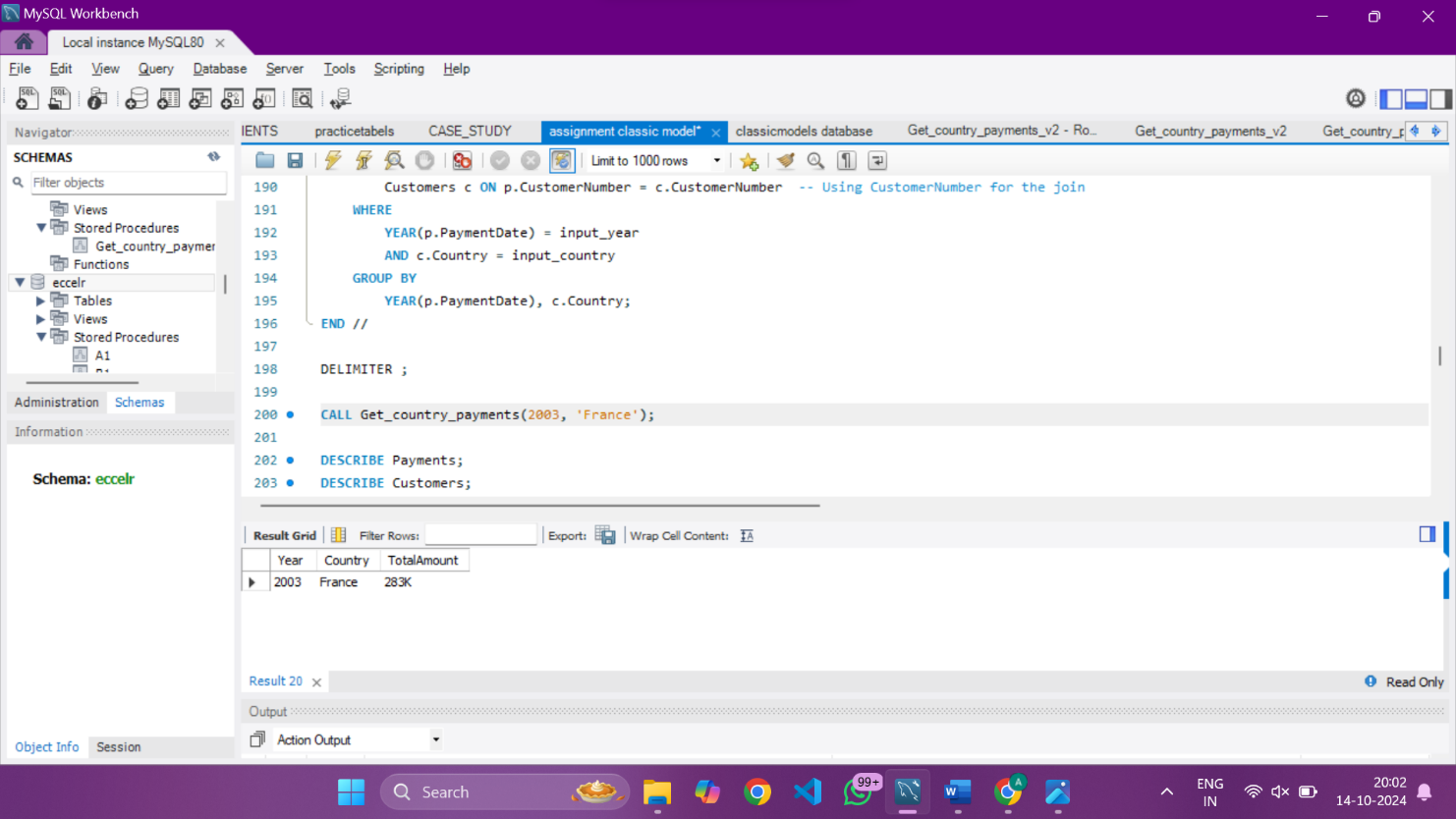
GROUP BY

YEAR(p.PaymentDate), c.Country;

END //

DELIMITER ;

CALL Get\_country\_payments(2003, 'France');



**Q10. Window functions - Rank, dense\_rank, lead and lag**

**a-answer**

SELECT c.customerName,

COUNT(o.orderNumber) AS Order\_count,

DENSE\_RANK() OVER (ORDER BY COUNT(o.orderNumber) DESC) AS order\_frequency\_rnk

FROM customers c

JOIN

orders o ON c.customerNumber = o.customerNumber

GROUP BY c.customerName

ORDER BY order\_frequency\_rnk;



**B-ANSWER---**

WITH MonthlyOrderCounts AS (

SELECT

YEAR(OrderDate) AS OrderYear,

MONTH(OrderDate) AS OrderMonth,

MONTHNAME(OrderDate) AS MonthName,

COUNT(OrderNumber) AS OrderCount

FROM

Orders

GROUP BY

OrderYear, OrderMonth

),

YoYChanges AS (

SELECT

OrderYear,

MonthName,

OrderMonth,

OrderCount,

Lag(OrderCount) OVER (PARTITION BY OrderYear) AS PreviousYearCount

FROM

MonthlyOrderCounts

)

SELECT

OrderYear,

MonthName,

OrderCount,

CASE

WHEN PreviousYearCount IS NULL THEN NULL

ELSE CONCAT(ROUND(((OrderCount - PreviousYearCount) / PreviousYearCount) \* 100), '%')

END AS YoYPercentageChange

FROM

YoYChanges

ORDER BY

OrderYear, OrderMonth;

**Q11.Subqueries and their applications**

**a-answer**

SELECT productLine, COUNT(\*) AS Total

FROM Products

WHERE buyPrice > (SELECT AVG(buyPrice)FROM Products)

GROUP BY productLine

ORDER BY Total desc**;**

****

**Q12. ERROR HANDLING in SQL**

**ANSWER-**

Create the table Emp\_EH. Below are its fields.

* EmpID (Primary Key)
* EmpName
* EmailAddress

CREATE TABLE Emp\_EH (

EmpID INT PRIMARY KEY,

EmpName VARCHAR(100),

EmailAddress VARCHAR(100)

);

DELIMITER //

CREATE PROCEDURE InsertEmp\_EH(

IN p\_EmpID INT,

IN p\_EmpName VARCHAR(100),

IN p\_EmailAddress VARCHAR(100)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

SELECT 'Error occurred';

ROLLBACK;

END;

START TRANSACTION;

INSERT INTO Emp\_EH (EmpID, EmpName, EmailAddress)

VALUES (p\_EmpID, p\_EmpName, p\_EmailAddress);

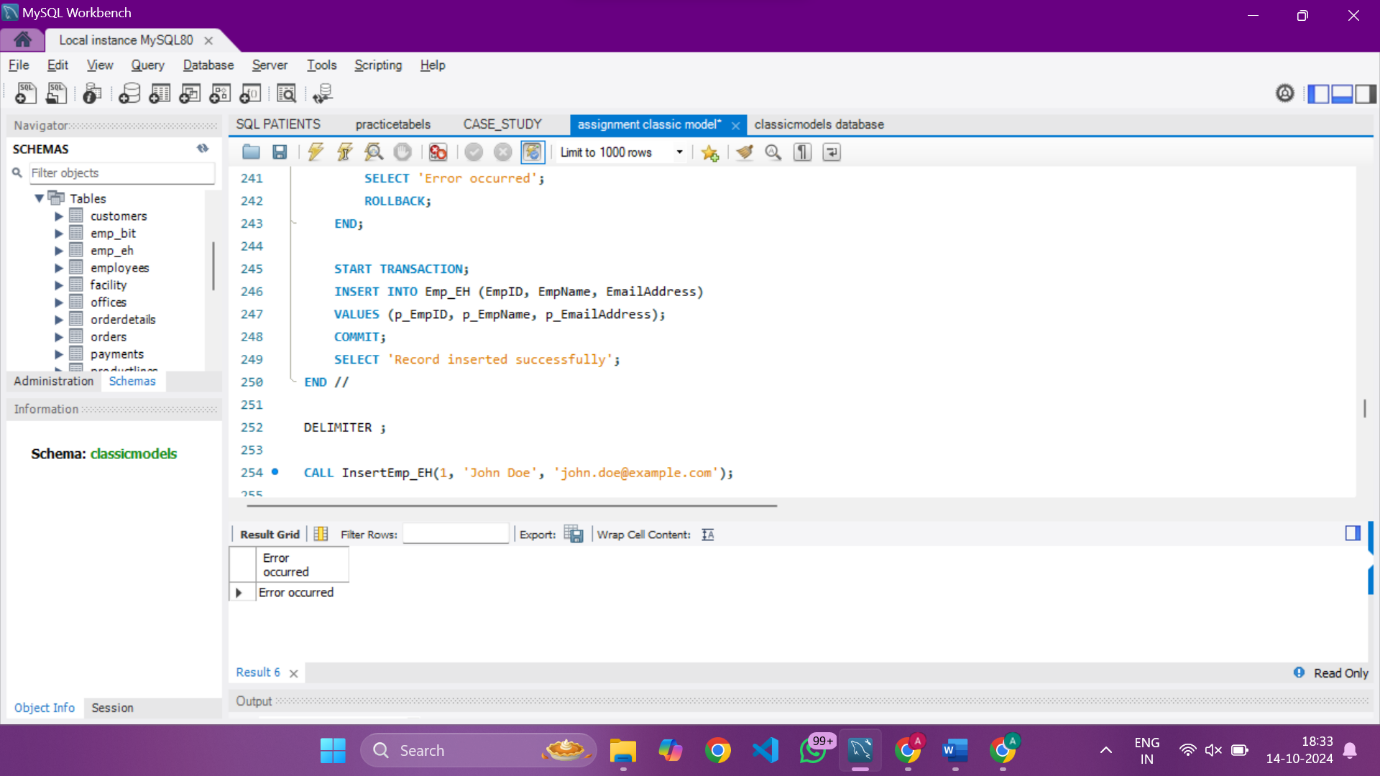
COMMIT;

SELECT 'Record inserted successfully';

END //

DELIMITER ;

CALL InsertEmp\_EH(1, 'John Doe', 'john.doe@example.com');



**Q13. TRIGGERS**

**Answer-**

# creatinğ the table

CREATE TABLE Emp\_BIT (

Name VARCHAR(100),

Occupation VARCHAR(100),

Working\_date DATE,

Working\_hours INT

);

select \* from emp\_bit;

#inserting the provide data

INSERT INTO Emp\_BIT VALUES

('Robin', 'Scientist', '2020-10-04', 12),

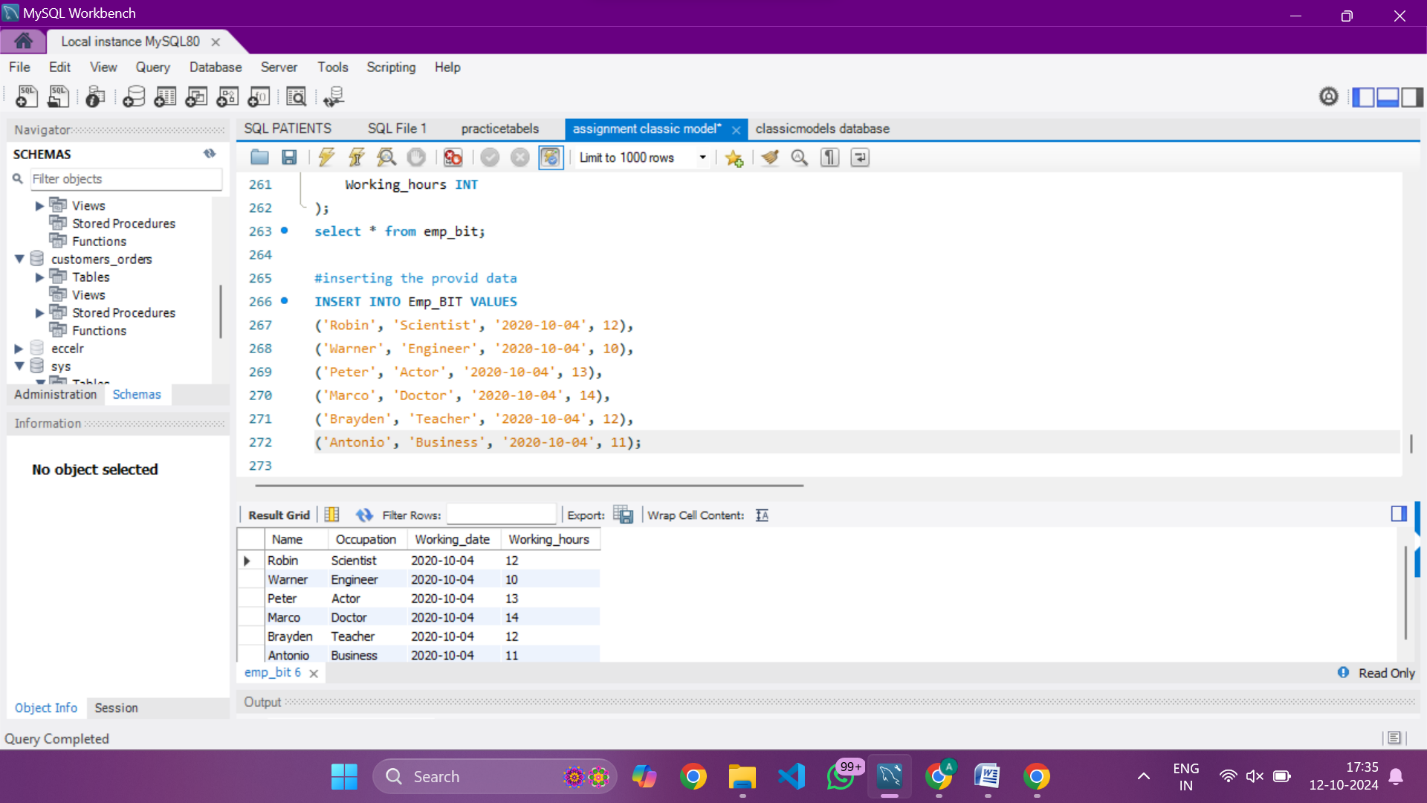
('Warner', 'Engineer', '2020-10-04', 10),

('Peter', 'Actor', '2020-10-04', 13),

('Marco', 'Doctor', '2020-10-04', 14),

('Brayden', 'Teacher', '2020-10-04', 12),

('Antonio', 'Business', '2020-10-04', 11);



**#CREATING THEBEFORE INSERT TRIGGER**

DELIMITER $$

CREATE TRIGGER Before\_Insert\_WorkingHours

BEFORE INSERT ON Emp\_BIT

FOR EACH ROW

BEGIN

IF NEW.Working\_hours < 0 THEN

SET NEW.Working\_hours = ABS(NEW.Working\_hours);

END IF;

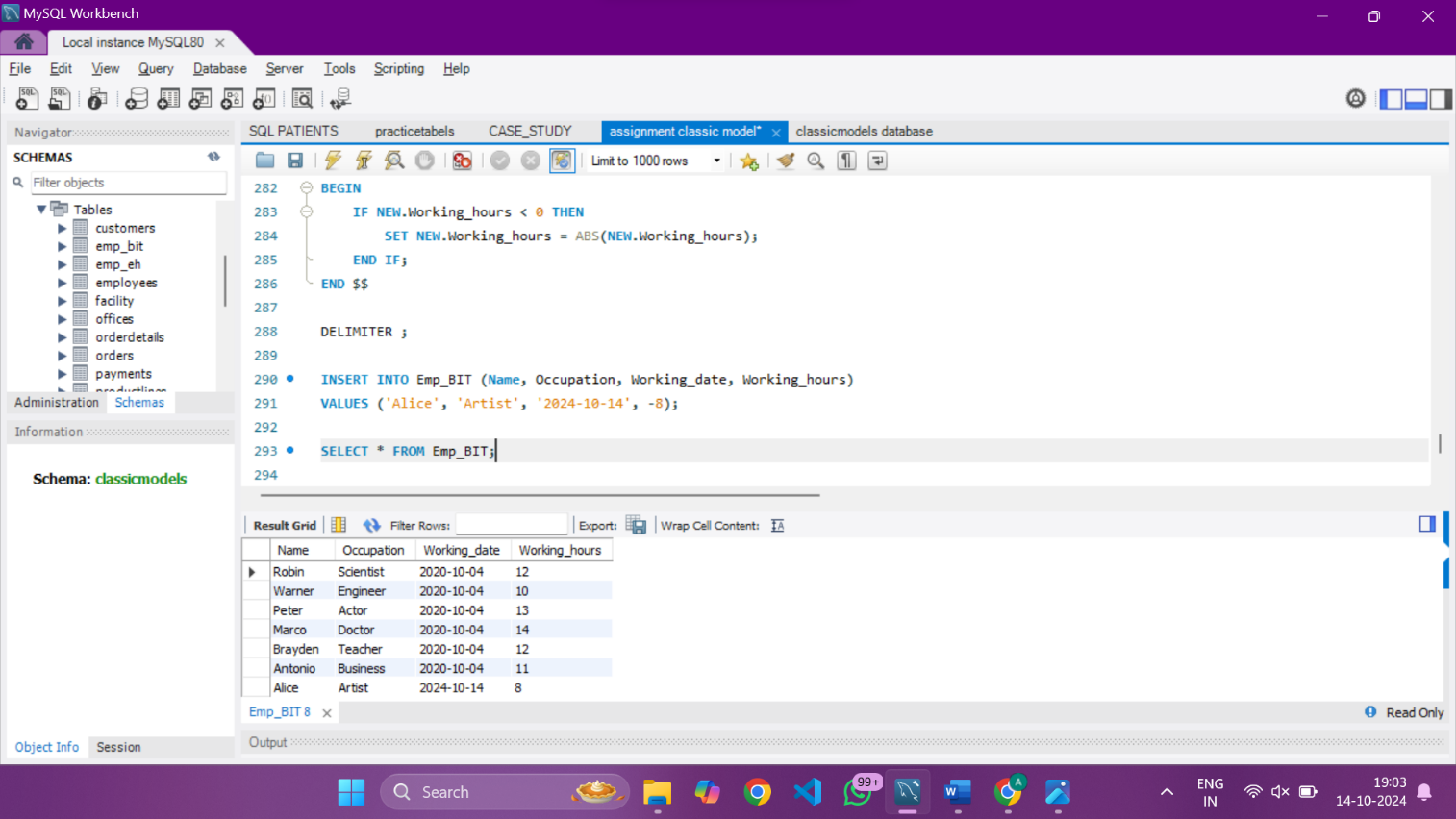
END $$

DELIMITER ;

INSERT INTO Emp\_BIT (Name, Occupation, Working\_date, Working\_hours)

VALUES ('Alice', 'Artist', '2024-10-14', -8);

SELECT \* FROM Emp\_BIT;



HERE IS MY SQL ASSIGNMENT

THANKYOU.