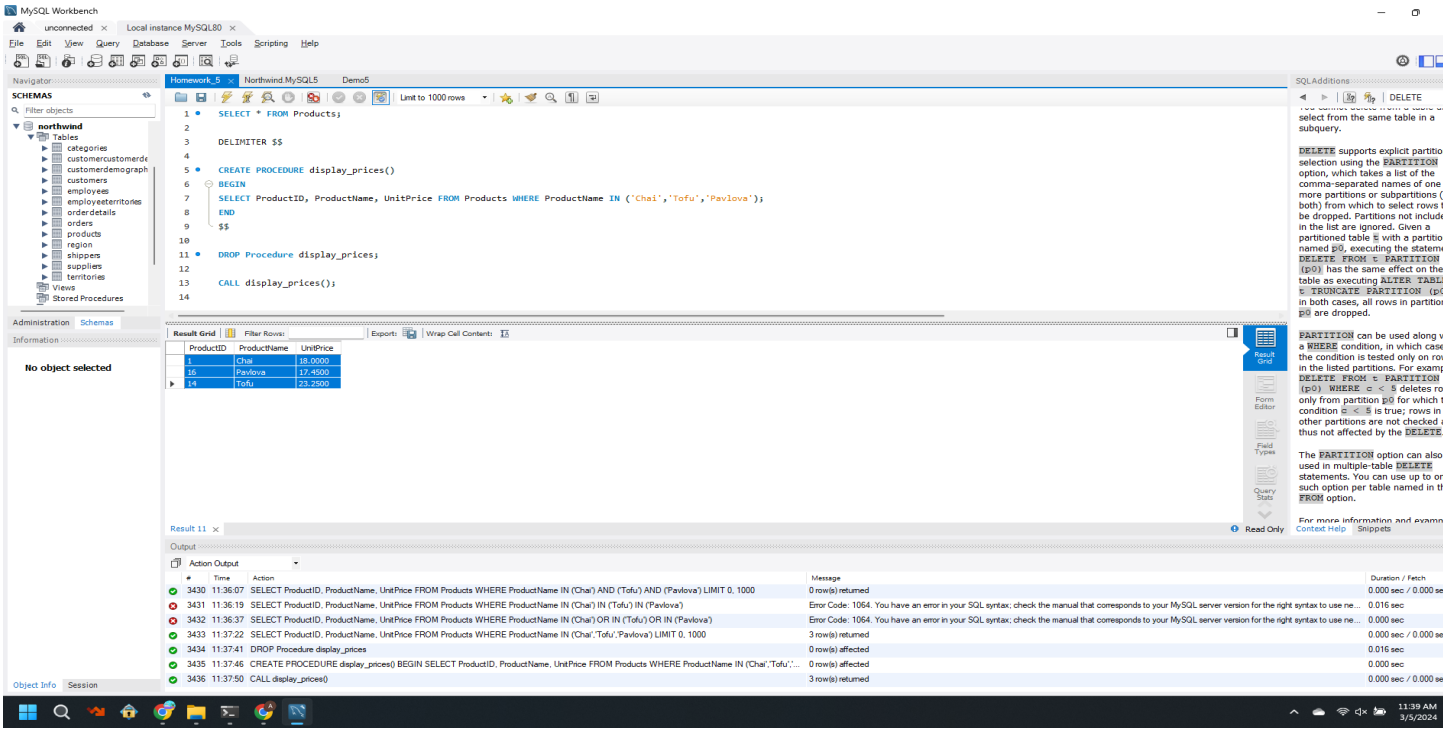
Homework_5_AparnaBharathi_Suresh

Question 1:

Write a stored procedure for Prices of Chai, Tofu, and Pavlova from the Products table in the **Northwind database**.

Screenshot 1:



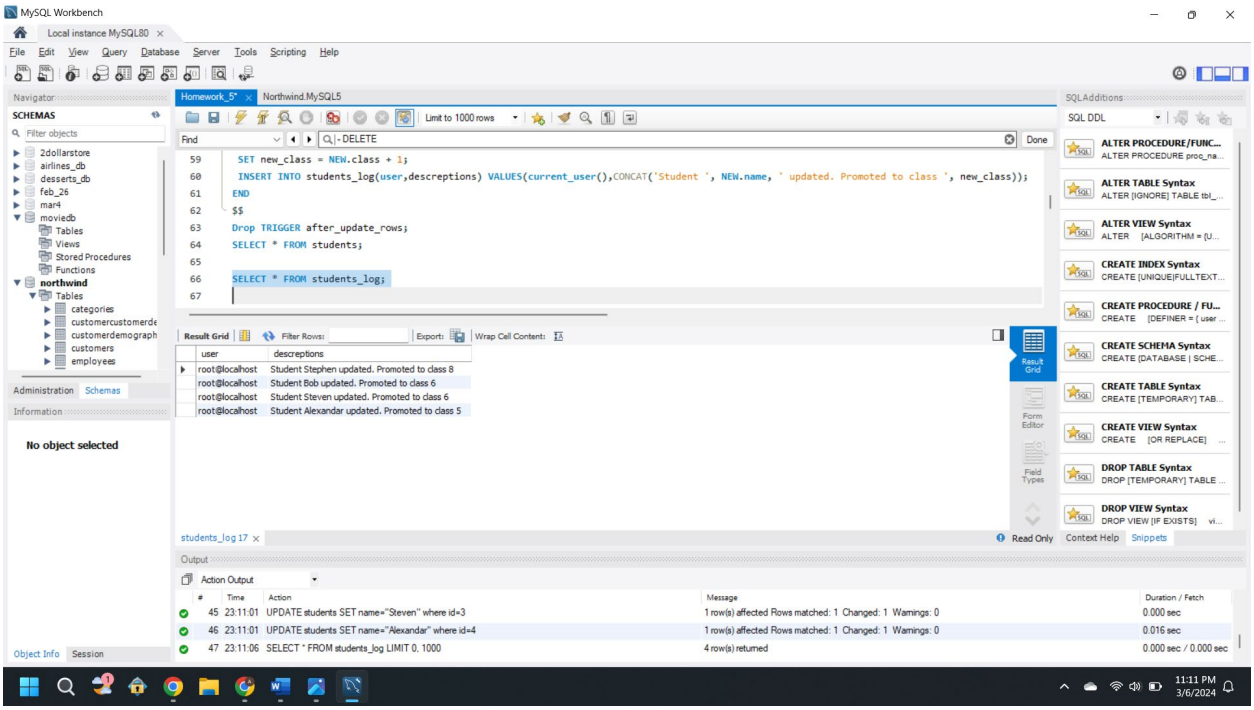
Query 1:

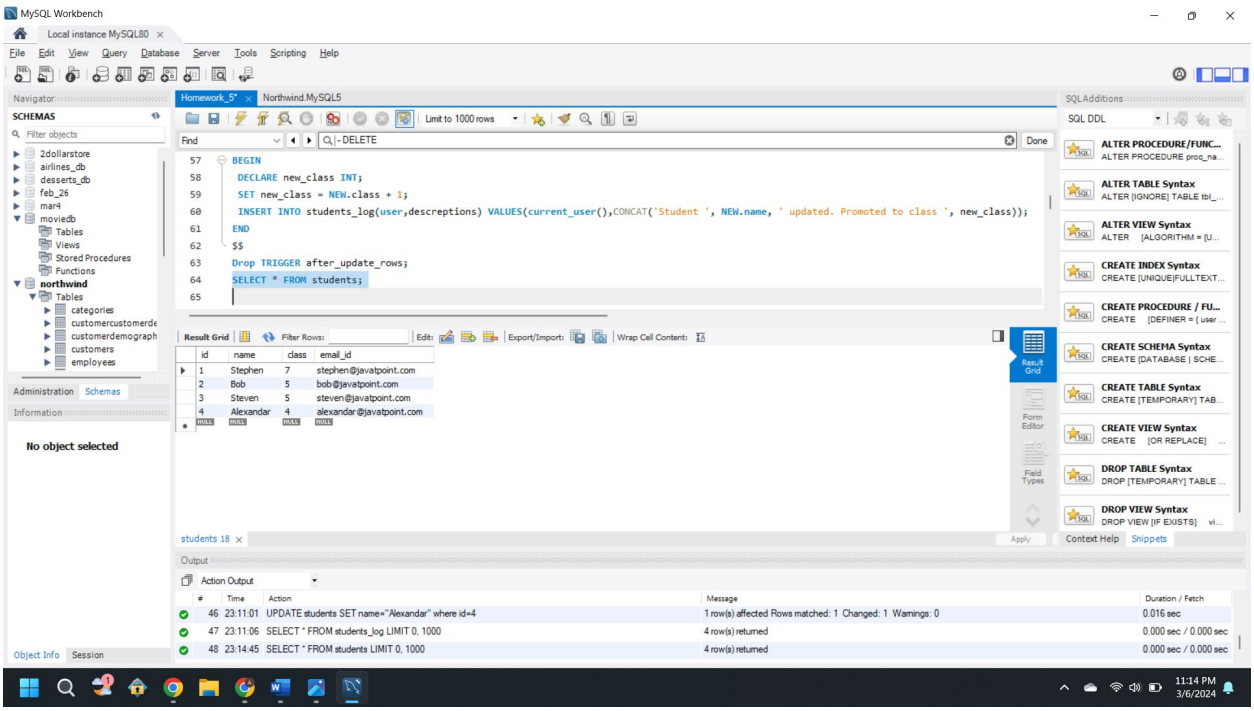
```
DELIMITER $$
CREATE PROCEDURE display_prices()
BEGIN
SELECT ProductID, ProductName, UnitPrice FROM Products WHERE ProductName IN
('Chai','Tofu','Pavlova');
END
$$
#Call the procedure:
CALL display_prices();
```

Question 2:

Create an AFTER UPDATE trigger that promotes all students in the next class, i.e., 6 will be 7, 7 will be 8, and so on. Whenever an update is performed on a single row in the "students" table, a new row will be inserted in the "students_log" table.

Screenshot 2:





Query 2:

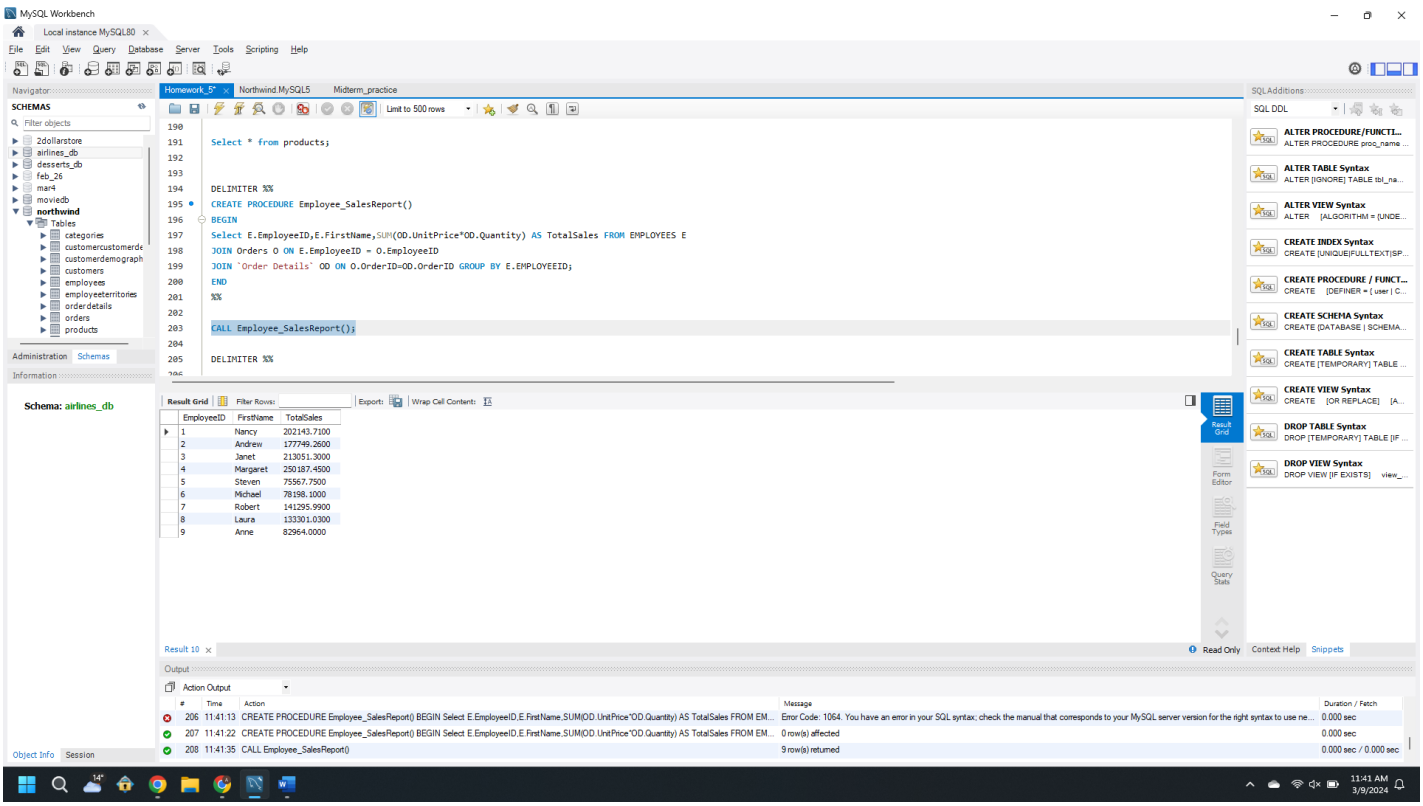
```
DELIMITER $$
CREATE TRIGGER after_update_rows
AFTER UPDATE
ON students FOR EACH ROW
BEGIN
 DECLARE new_class INT;
 SET new_class = NEW.class + 1;
 INSERT INTO students_log(user,descreptions) VALUES(current_user(),CONCAT('Student ',
NEW.name, ' updated. Promoted to class ', new_class));
END
$$
```

Question 3:

Write 1 **useful** stored procedure and trigger using the **Northwind Database**. (No restrictions for the type of stored procedure and trigger)

Screenshot 3:

<u>Procedure:</u>
I created an employee sales report Procedure which displays the EmployeeID, Name, and the Total Sales by each employee that is Sum(UnitPrice*quantity).



Query 3:

DELIMITER %%
CREATE PROCEDURE Employee_SalesReport()
BEGIN
Select E.EmployeeID,E.FirstName,SUM(OD.UnitPrice*OD.Quantity) AS TotalSales FROM EMPLOYEES E

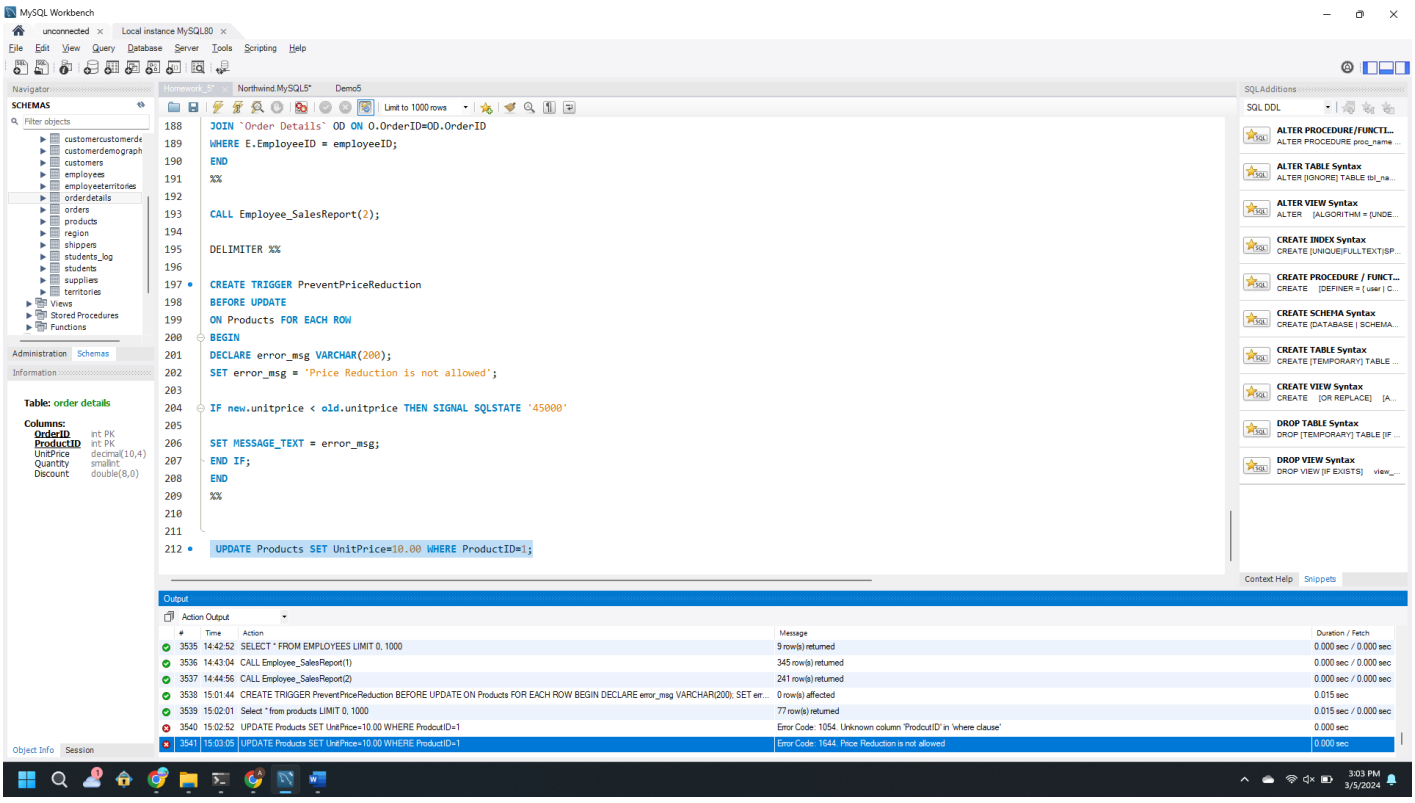
JOIN Orders O ON E.EmployeeID = O.EmployeeID
JOIN `Order Details` OD ON O.OrderID=OD.OrderID GROUP BY E.EMPLOYEEID;
END
%%

CALL Employee_SalesReport();

Screenshot 3:

<u>Trigger:</u>

I created a trigger to prevent price reduction for each product. If you update the unit price of a product to a lesser value than the actual value, then it will throw error stating that Price reduction is not allowed.



Query 3:

DELIMITER %%

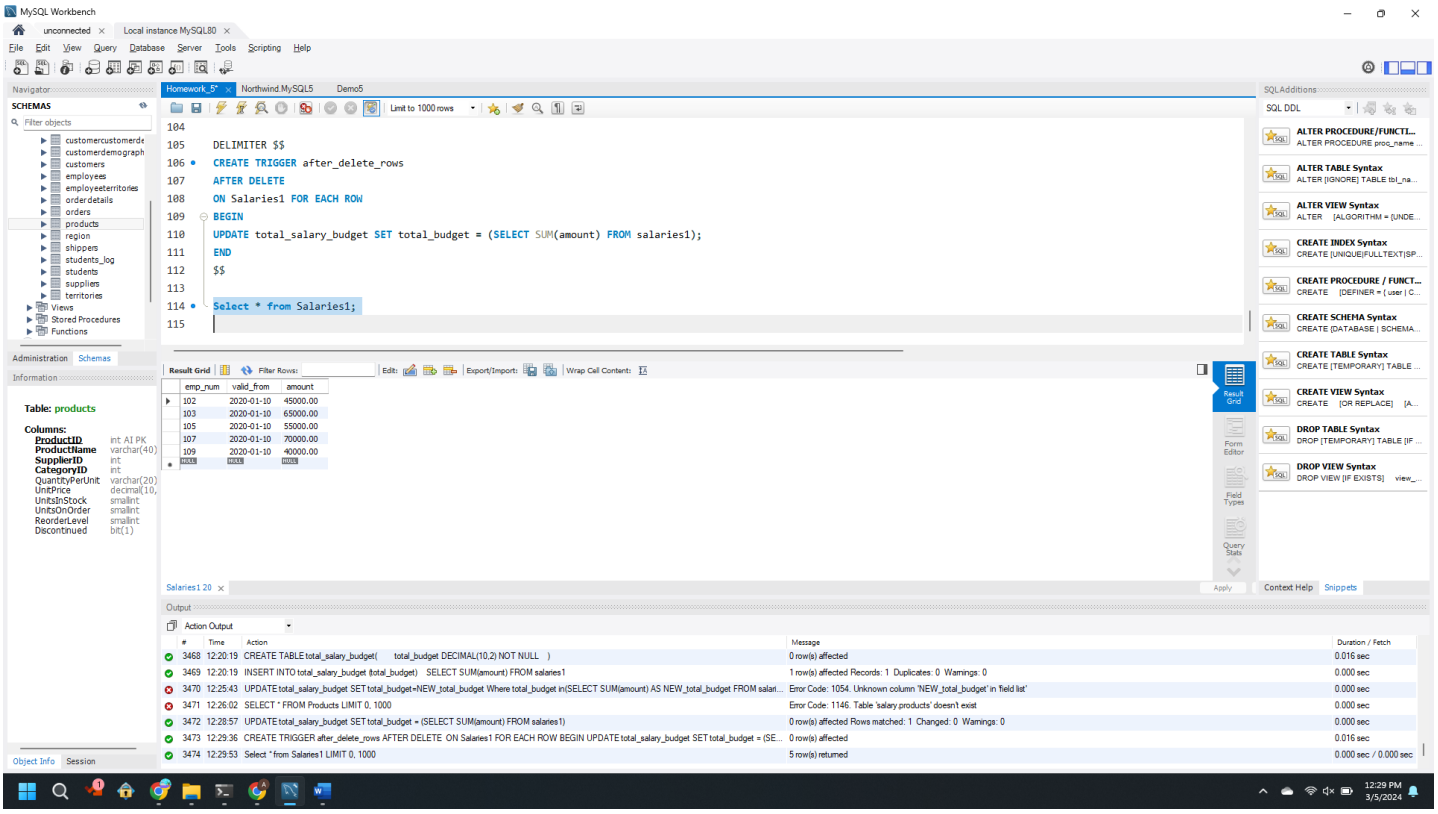
```
CREATE TRIGGER PreventPriceReduction
BEFORE UPDATE
ON Products FOR EACH ROW
BEGIN
DECLARE error_msg VARCHAR(200);
SET error_msg = 'Price Reduction is not allowed';

IF new.unitprice < old.unitprice THEN SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = error_msg;
END IF;
END
%%

#Update Query:
 UPDATE Products SET UnitPrice=10.00 WHERE ProductID=1;
```

Question 4:

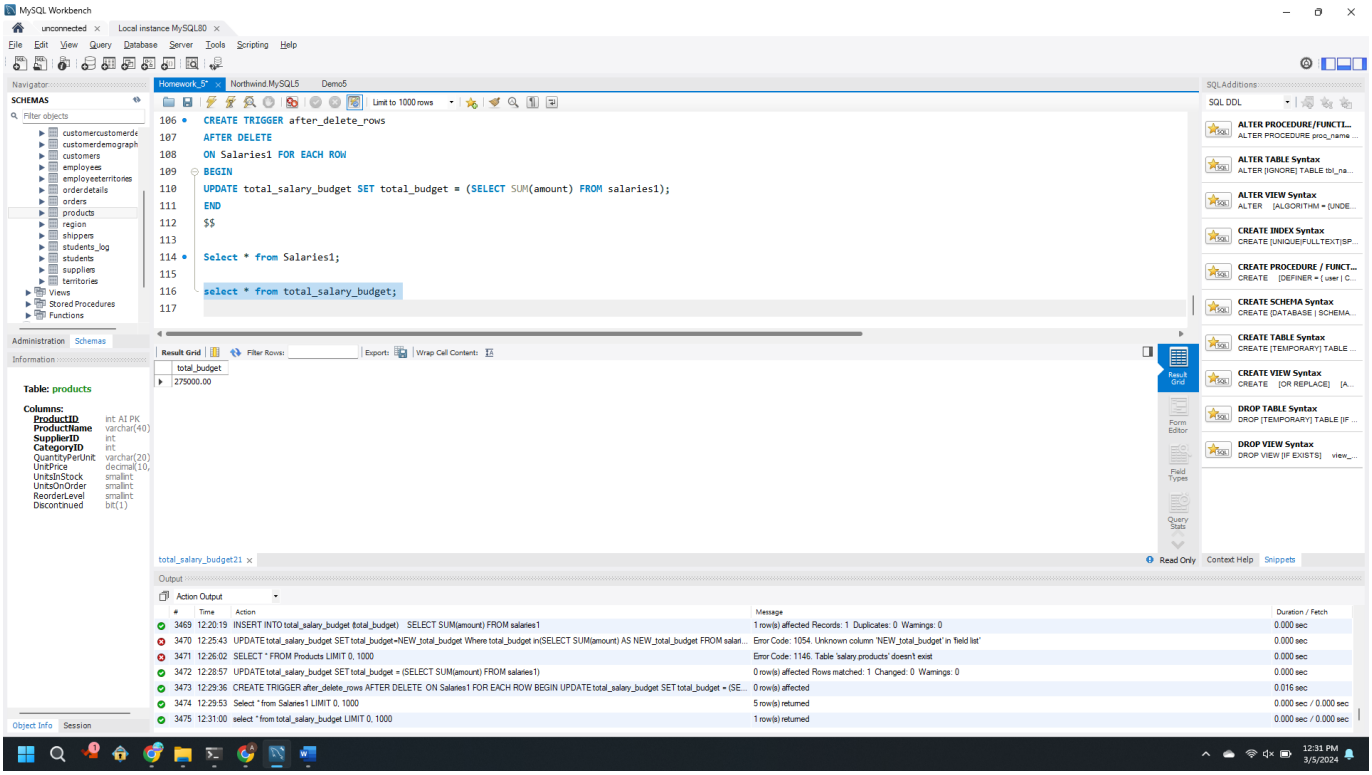
Write the trigger logic that updates the total salary into the total_salary_budget table after a row is deleted from the salaries table.

Screenshot 4:

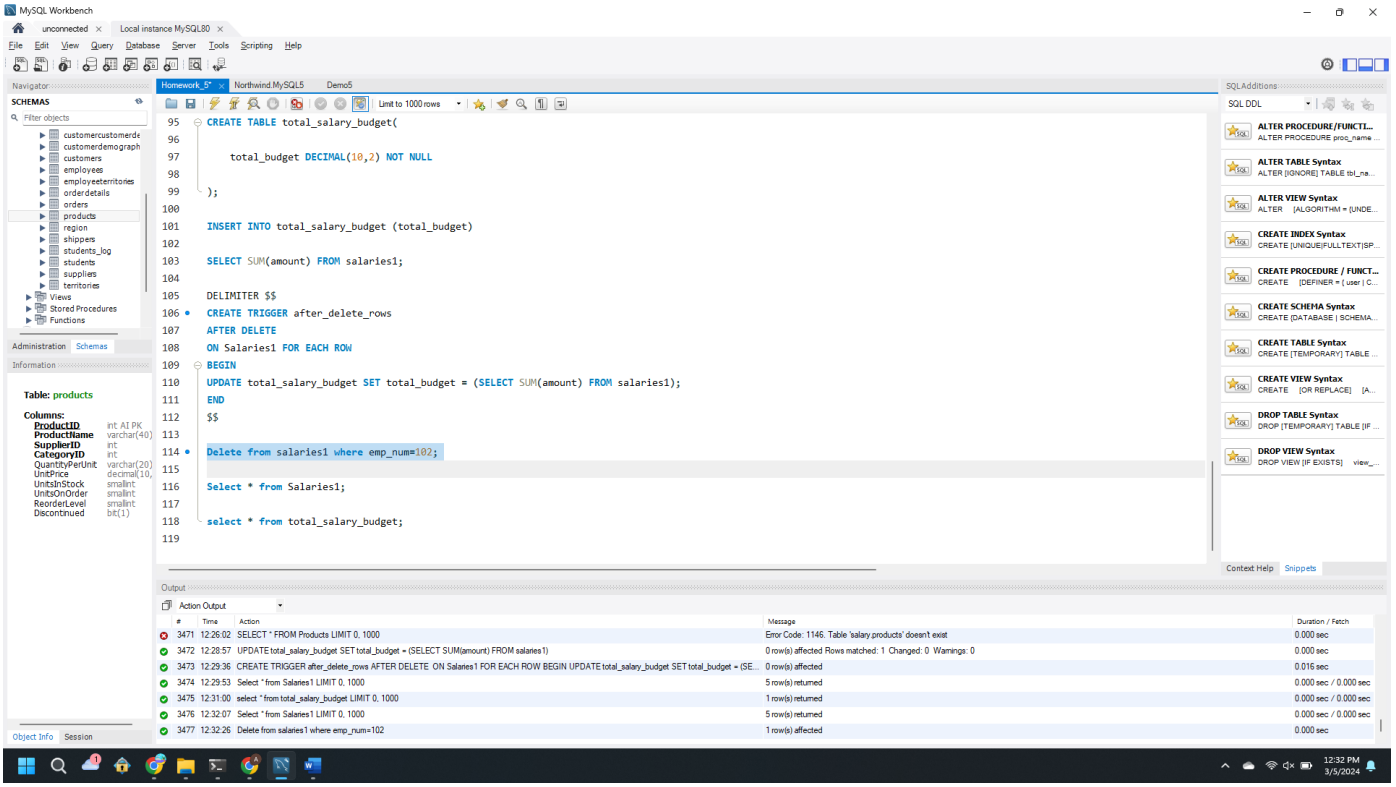
Before Delete Salaries table:



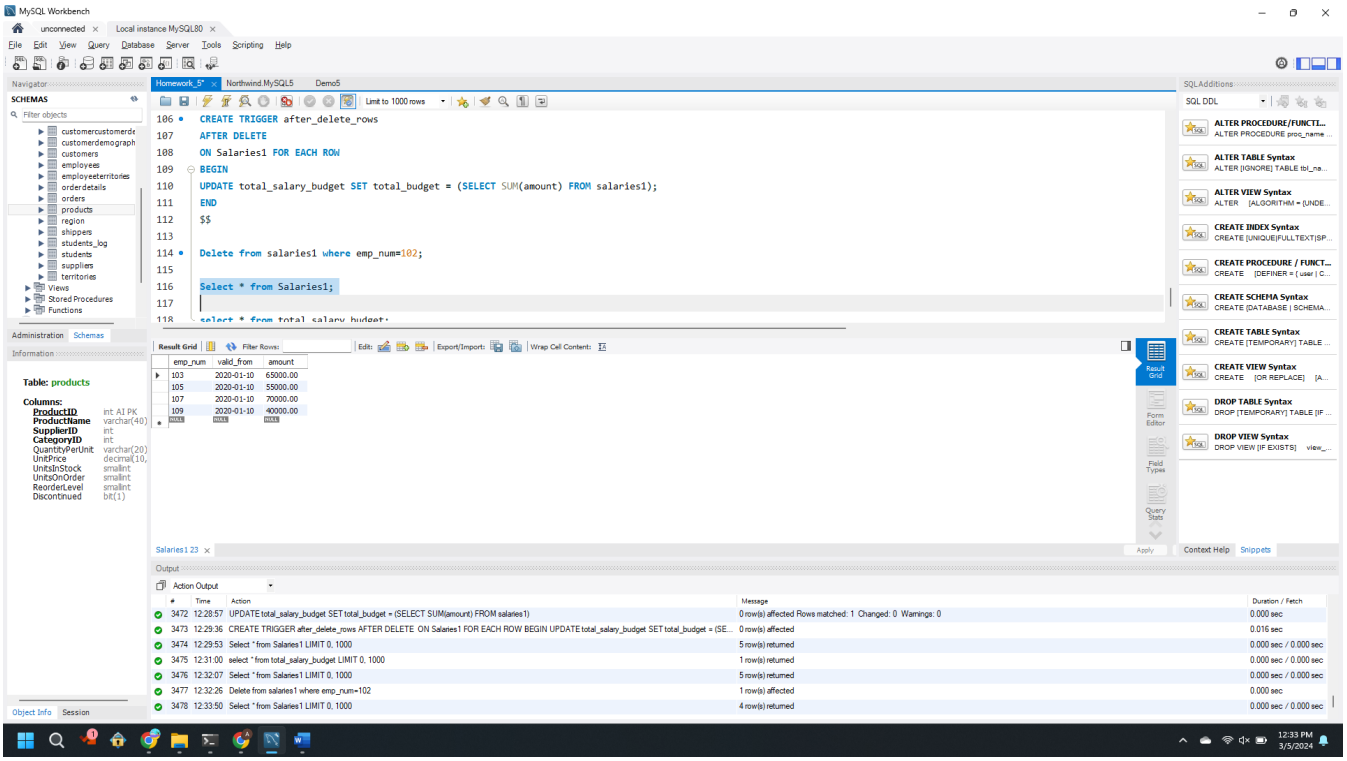
Before delete total salary budget table:



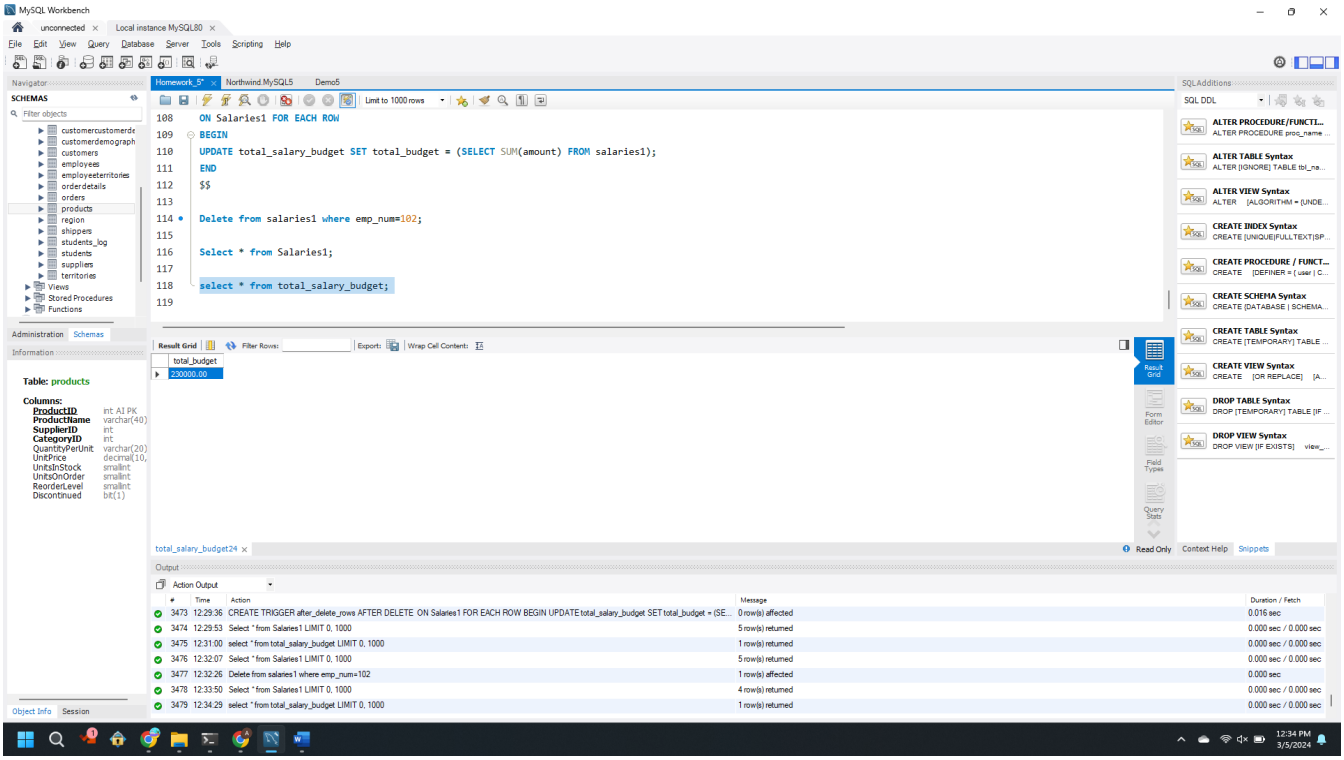
Delete a row:



Salary table after deleting:



Salary budget table after deleting:



Query 4:

#Trigger:

```
DELIMITER $$
CREATE TRIGGER after_delete_rows
AFTER DELETE
ON Salaries1 FOR EACH ROW
BEGIN
UPDATE total_salary_budget SET total_budget = (SELECT SUM(amount) FROM salaries1);
END
$$
```

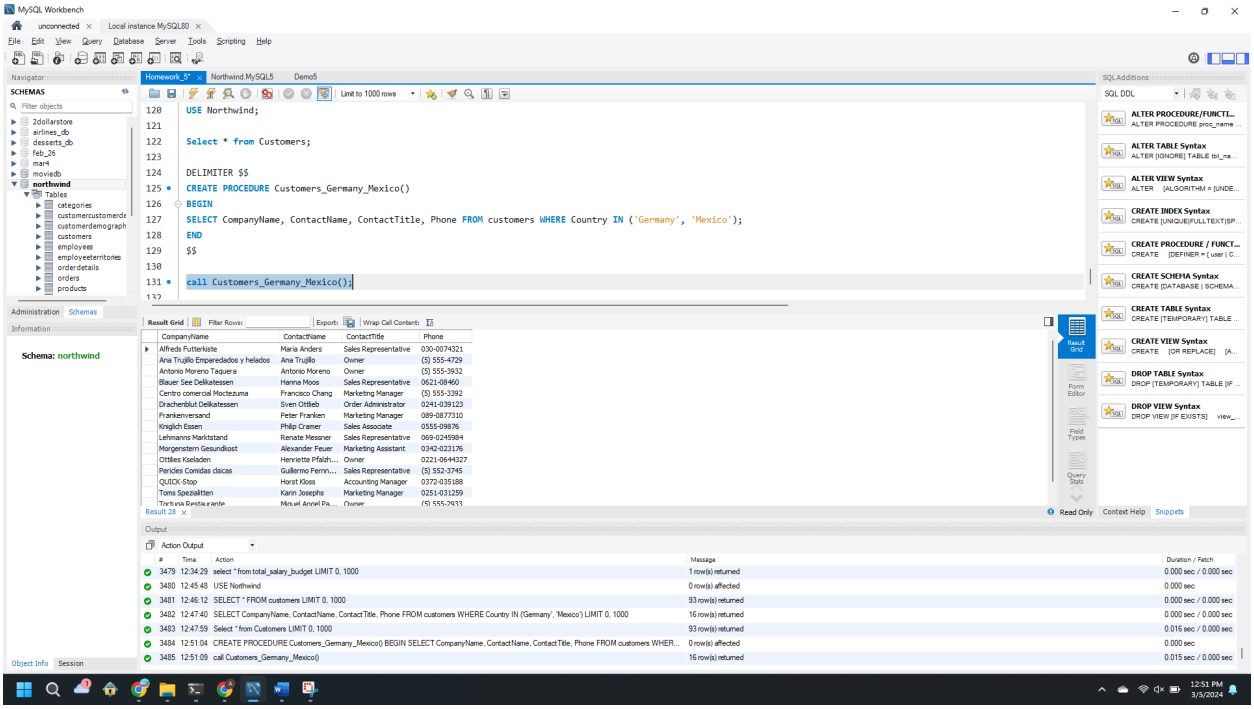
#Delete Query

```
DELETE FROM salaries1 where emp_num=102;
```

Question 5:

Show customers(CompanyName,ContactName,ContactTitle,Phone) from Germany and Mexico using stored procedures.

Screenshot 5:



Query 5:

#Procedure:

DELIMITER $$

CREATE PROCEDURE Customers_Germany_Mexico()

BEGIN

SELECT CompanyName, ContactName, ContactTitle, Phone FROM customers WHERE Country IN ('Germany', 'Mexico');

END

$$

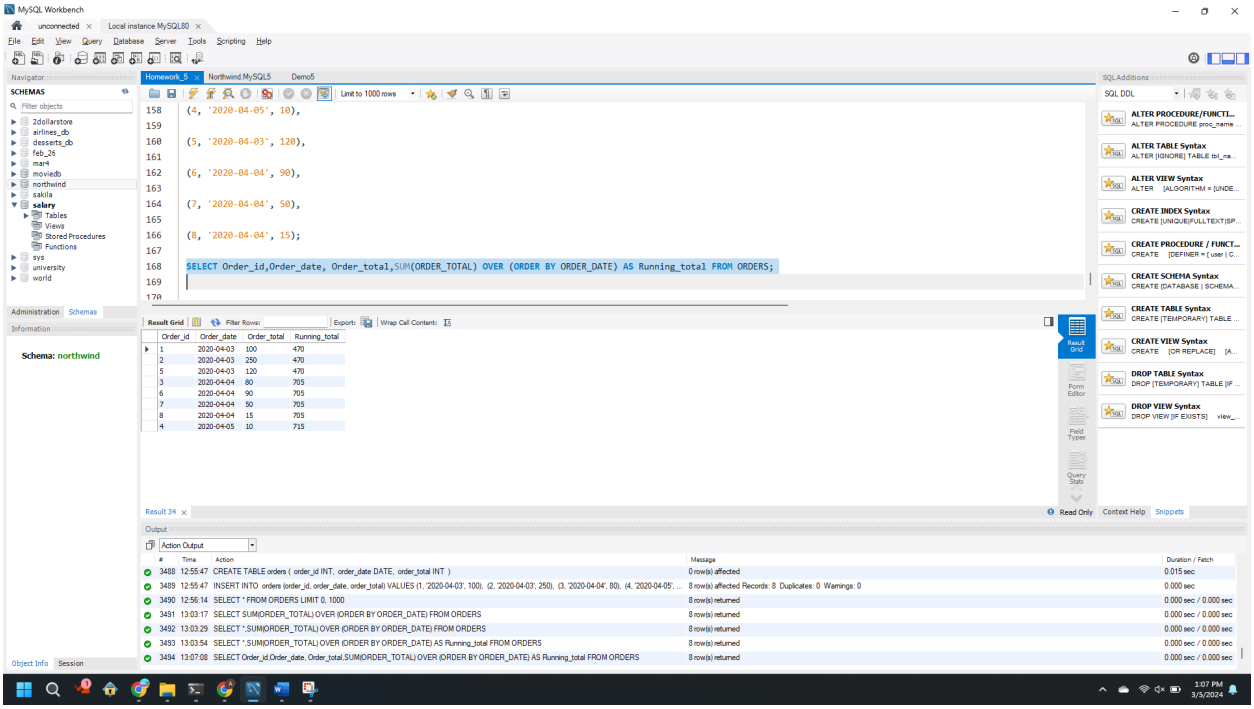
#Call Procedure:

call Customers_Germany_Mexico();

Question 6:

Using the schema below, write a query to display order id, order date, order total and running order total based on the order date, and sort the results using order date. (2 marks)

Screenshot 6:



Query 6:

SELECT Order_id,Order_date, Order_total,SUM(Order_Total) OVER (ORDER BY Order_Date) AS Running_total FROM Orders;