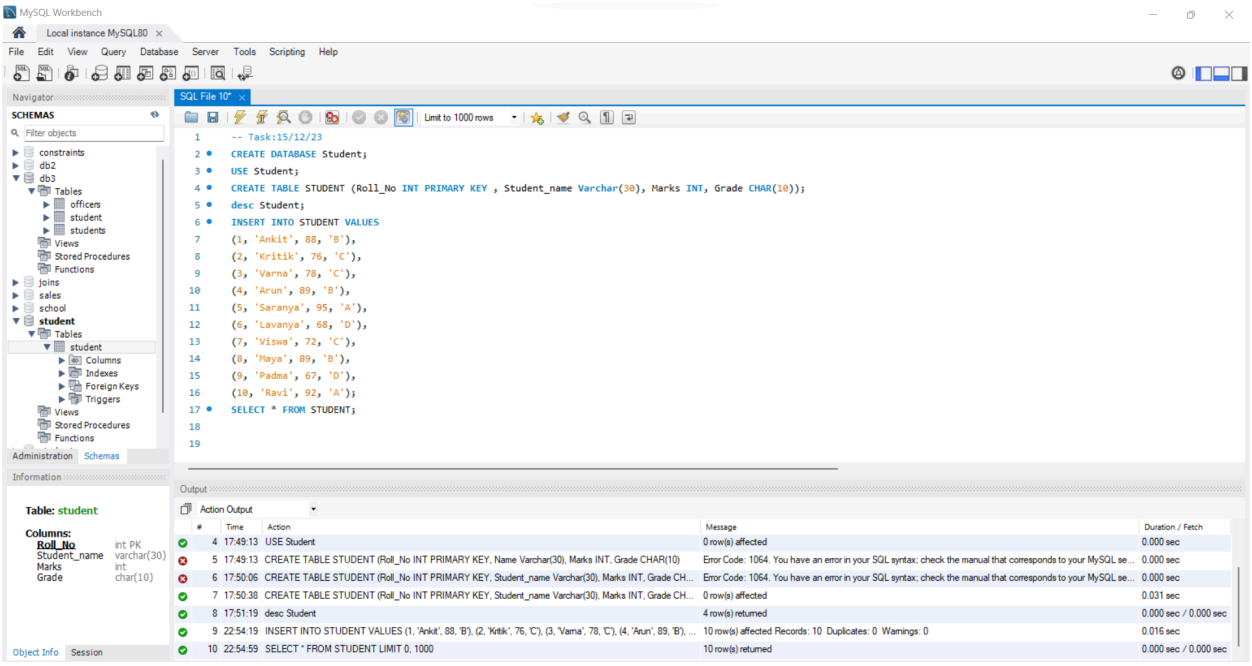
15/12/2023

Create a database named School and perform all the DDL commands(CREATE, ALTER, DROP, RENAME, TRUNCATE) for the table named STUDENT with fields:

Roll.No
Name
Marks
Grade

(a) Insert 10 rows into the table using INSERT INTO.

(b) Use the select command to display the table.



(c) Add a column named Contact to the STUDENT table.
(d) Remove the Grade column from the Student table.
(e) Rename the table to CLASSTEN.
(f) Delete all rows from the table.
(g) Remove the table from the database.

```
16      (10, 'Ravi', 92, 'A');
17 •    SELECT * FROM STUDENT;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Roll_No	Student_name	Marks	Grade
1	Ankit	88	B
2	Kritik	76	C
3	Varna	78	C
4	Arun	89	B
5	Saranya	95	A
6	Lavanya	68	D
7	Viswa	72	C
8	Maya	89	B
9	Padma	67	D
10	Ravi	92	A

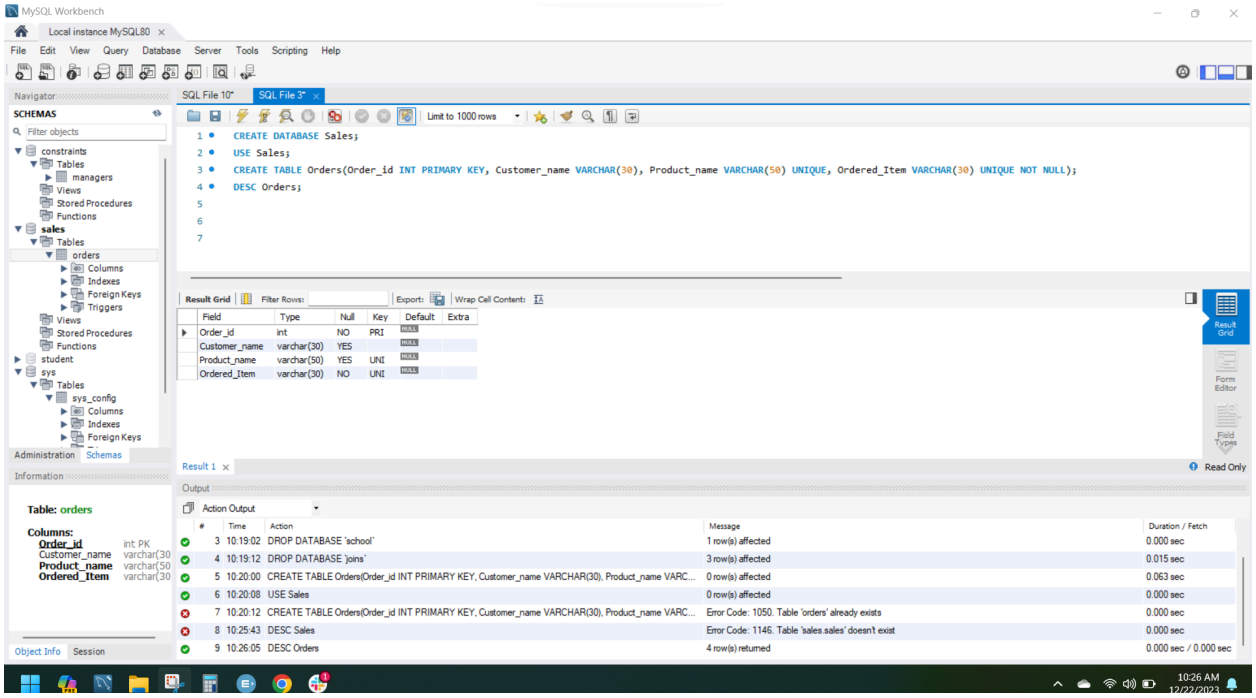
TUDENT 3

****

19/12/23

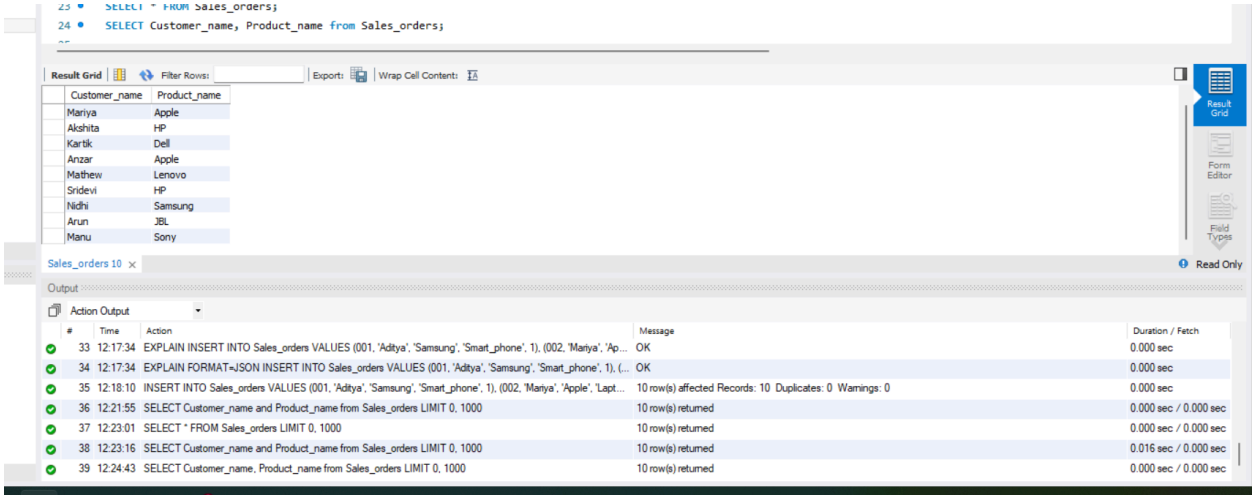
Create a database called “Sales” and create a new table named “Orders” in the Sales database with columns: (Order_Id, Customer_name, Product_name, Ordered_item). Use constraints:

- **Primary Key**
- **Unique**
- **Not Null**

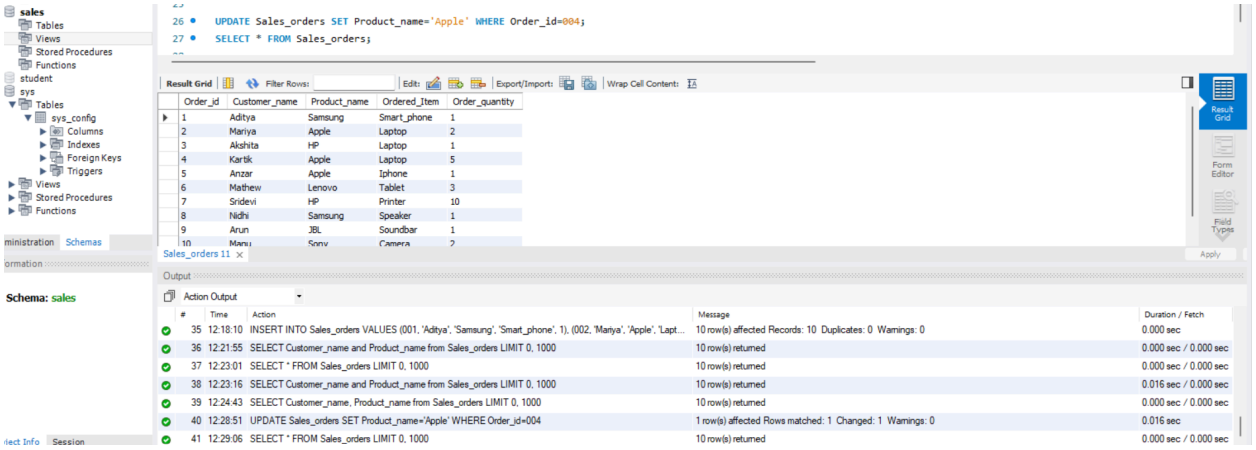


1.Add a new column named “order_quantity” to the orders table.
2.Rename the orders table to the sales_orders table.
3. Insert 10 rows into the sales_orders table.

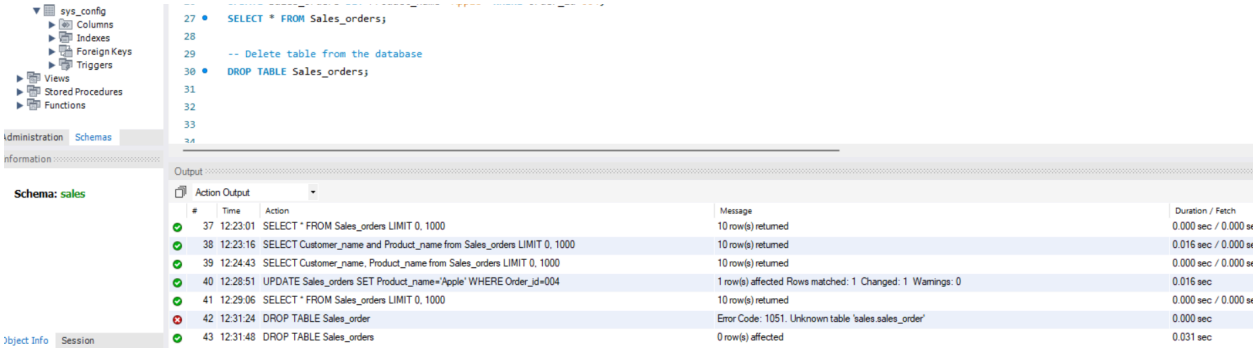
4.Retrieve customer_name and Product_name from the sales_orders table.



5.Use the update command to change the name of the product for any row.



6. Delete the sales_orders table from the database.



5/01/23

Task 1[Joins]

Questions:

Insert 10 rows to both tables given below.

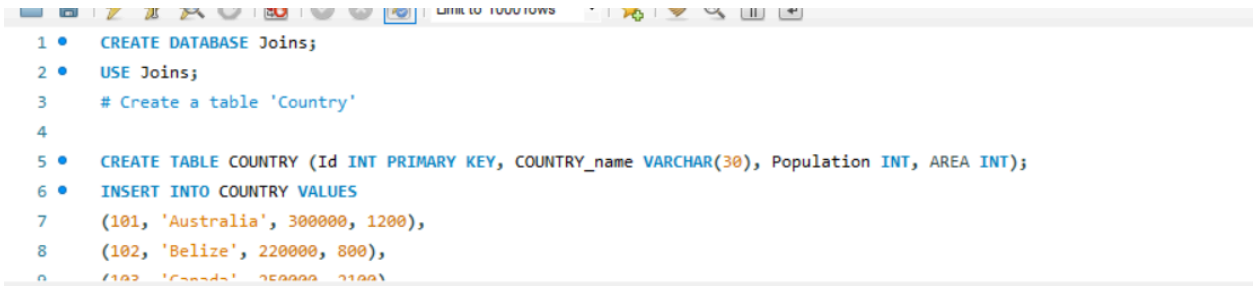
Create a table named Country with fields:

- **Id**
- **Country_name**
- **Population**
- **Area**

Create another table named Persons with fields:

- **Id**
- **Fname**
- **Lname**
- **Population**
- **Rating**
- **Country_Id**
- **Country_name**

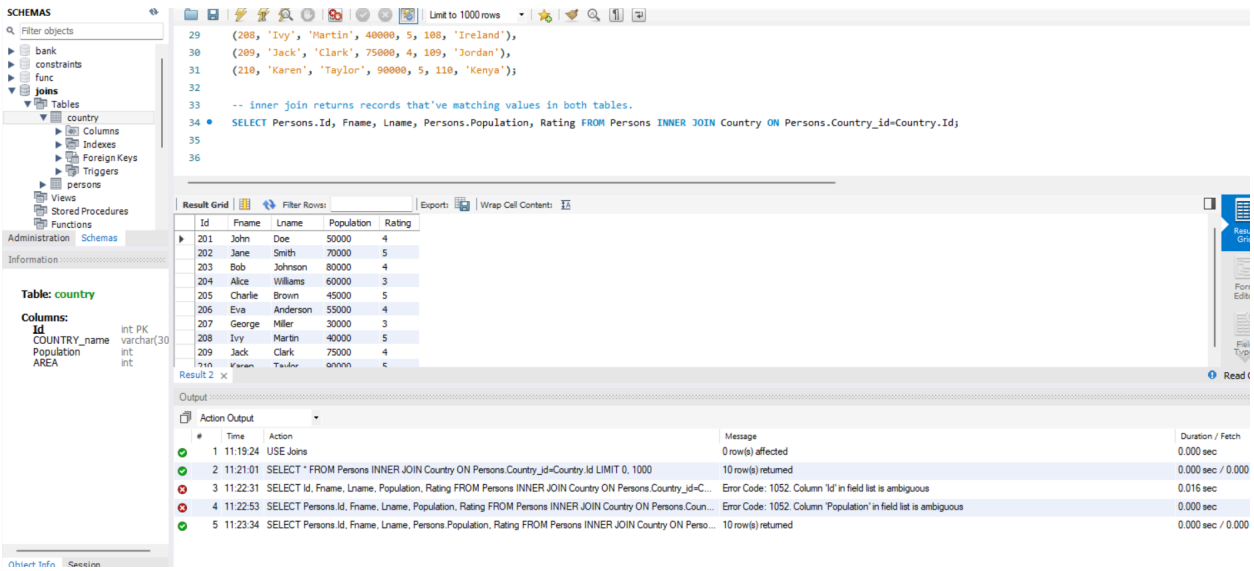
Perform inner join, Left join, and Right join on the tables above.



```
1  CREATE DATABASE Joins;
2  USE Joins;
3  # Create a table 'Country'
4
5  CREATE TABLE COUNTRY (Id INT PRIMARY KEY, COUNTRY_name VARCHAR(30), Population INT, AREA INT);
6  INSERT INTO COUNTRY VALUES
7  (101, 'Australia', 300000, 1200),
8  (102, 'Belize', 220000, 800),
```

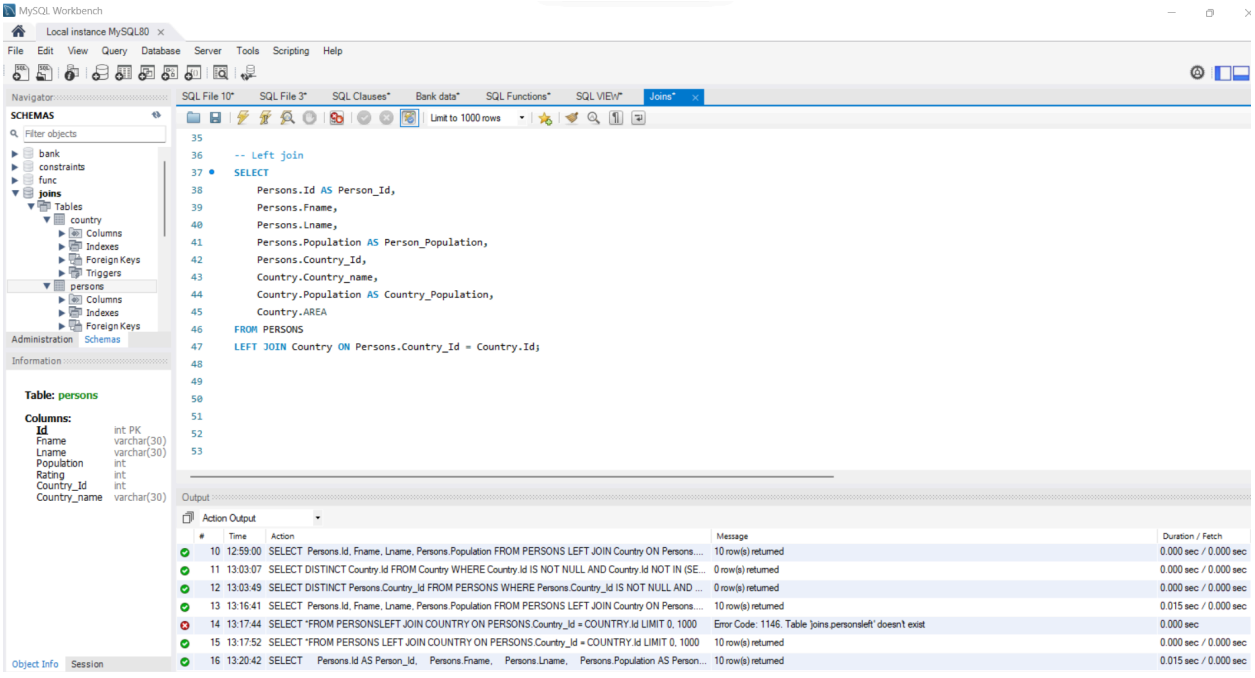
```sql
# 'Create a table Persons

CREATE TABLE PERSONS(Id INT PRIMARY KEY, Fname VARCHAR(30), Lname VARCHAR(30), Population INT, Rating INT, Country_Id INT, Country_name VARCHAR(30));
INSERT INTO PERSONS VALUES
(201, 'John', 'Doe', 50000, 4, 101, 'Australia'),
(202, 'Jane', 'Smith', 70000, 5, 102, 'Belize'),
(203, 'Bob', 'Johnson', 80000, 4, 103, 'Canada'),
(204, 'Alice', 'Williams', 60000, 3, 104, 'Denmark'),
```

Inner join



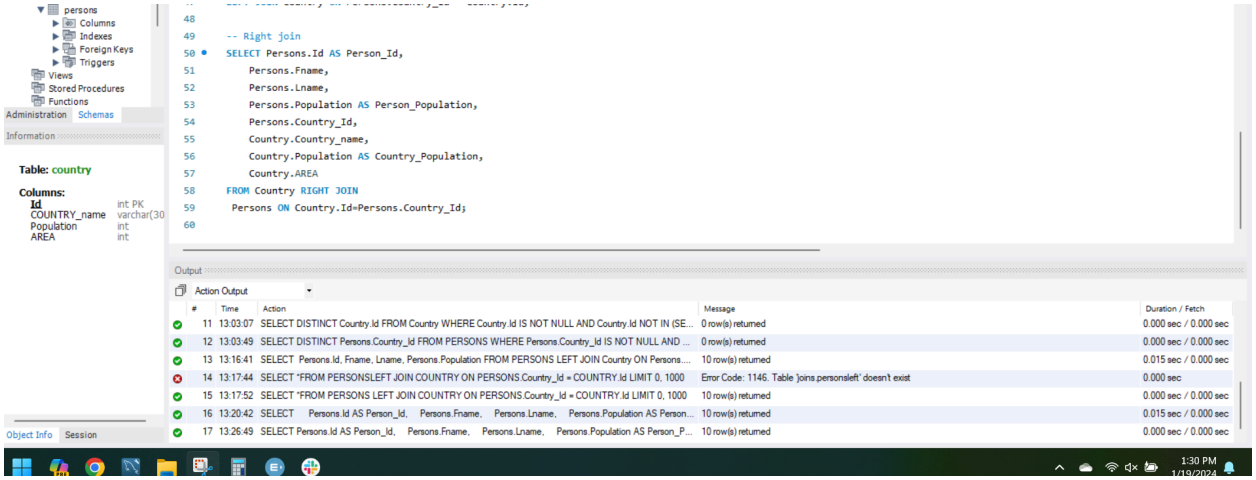
```sql
(208, 'Ivy', 'Martin', 40000, 5, 108, 'Ireland'),
(209, 'Jack', 'Clark', 75000, 4, 109, 'Jordan'),
(210, 'Karen', 'Taylor', 90000, 5, 110, 'Kenya');

-- inner join returns records that've matching values in both tables.
SELECT Persons.Id, Fname, Lname, Persons.Population, Rating FROM Persons INNER JOIN Country ON Persons.Country_id=Country.Id;
```

Left join



```sql
-- Left join
SELECT
    Persons.Id AS Person_Id,
    Persons.Fname,
    Persons.Lname,
    Persons.Population AS Person_Population,
    Persons.Country_Id,
    Country.Country_name,
    Country.Population AS Country_Population,
    Country.AREA
FROM PERSONS
LEFT JOIN Country ON Persons.Country_Id = Country.Id;
```

Right join

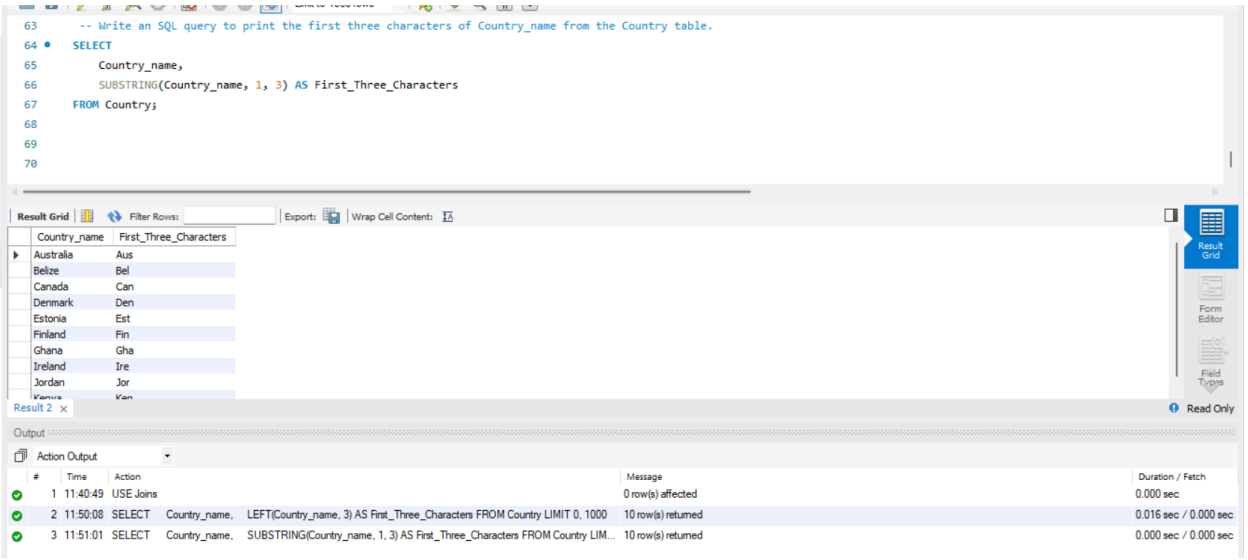


Task 2:[Built-in functions]

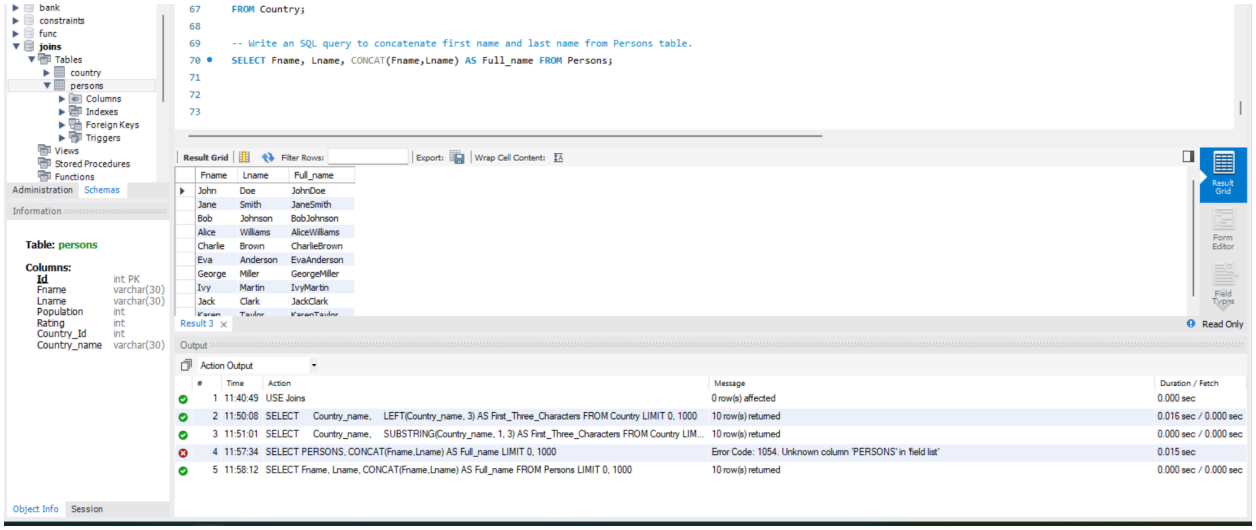
Questions:

Consider the Country table and Persons table that you created earlier.

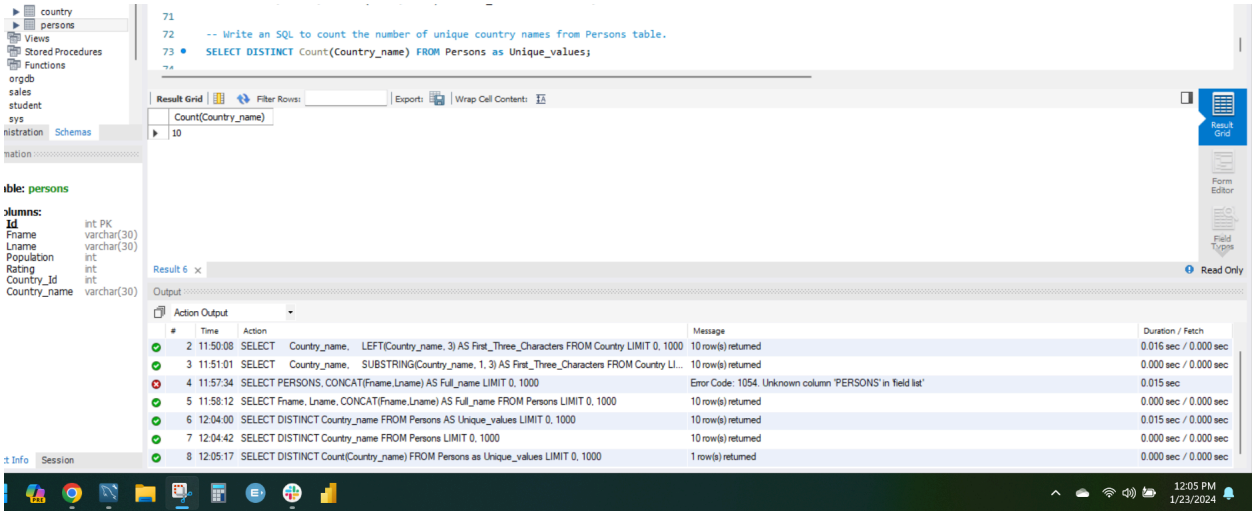
1. **Write an SQL query to print the first three characters of Country_name from the Country table.**



2. Write an SQL query to concatenate first name and last name from Persons table.



3. Write an SQL to count the number of unique country names from Persons table.



4. Write a query to print the maximum population from the Country table.

5.Write a query to print the minimum population from Persons table.

```sql
-- Write a query to print the maximum population from the Country table.
SELECT MAX(Population) AS Max_Population FROM Country;
-- Write a query to print the minimum population from the Country table.
SELECT MIN(Population) AS Min_Population FROM Country;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Min_Population
110000

Result 8 | Result 9 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
6	12:04:00	SELECT DISTINCT Country_name FROM Persons AS Unique_values LIMIT 0, 1000	10 row(s) returned	0.015 sec / 0.000 sec
7	12:04:42	SELECT DISTINCT Country_name FROM Persons LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
8	12:05:17	SELECT DISTINCT Count(Country_name) FROM Persons as Unique_values LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
9	12:17:18	SELECT Population Max(Population) AS Max_Population FROM Country	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
10	12:17:56	SELECT MAX(Population) AS Max_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
11	12:19:06	SELECT MAX(Population) AS Max_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
12	12:19:06	SELECT MIN(Population) AS Min_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

6. Insert 2 new rows to the Persons table making the Lname NULL. Then write another query to count Lname from Persons table.

```sql
-- Insert 2 new rows to the Persons table making the Lname NULL. Then write another query to count Lname from Persons table.
INSERT INTO Persons VALUES
(215, 'Miriam', Null, 45000, 5, 105, 'Estonia'),
(216, 'Nate', Null, 40000, 5, 108, 'Ireland');
SELECT COUNT(Lname) AS Count_Lname FROM Persons;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Count_Lname
10

Result 10

Output

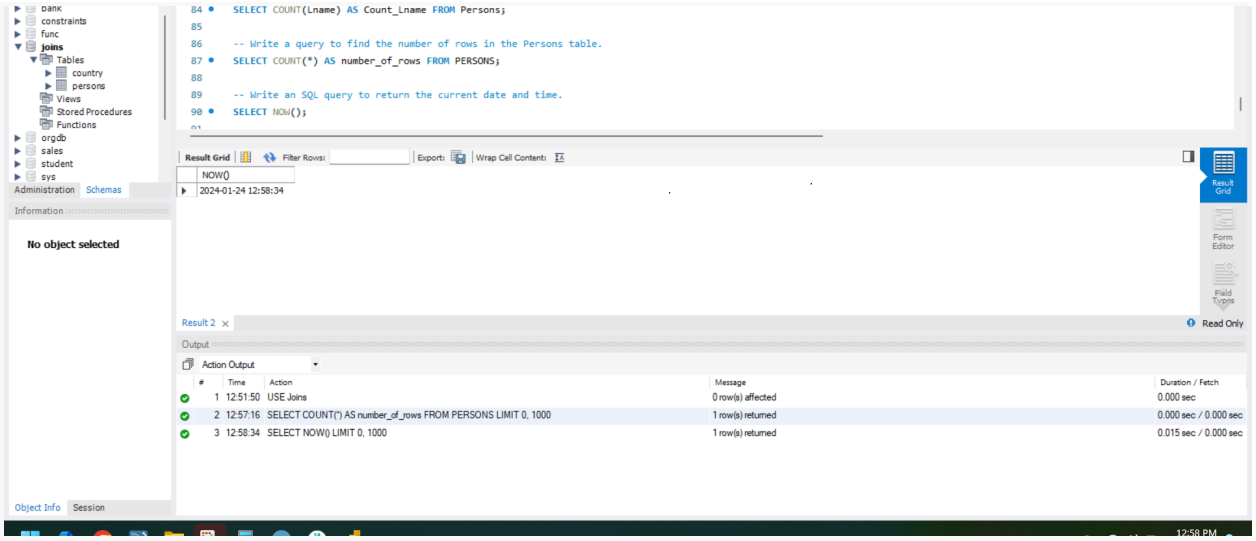
Action Output

#	Time	Action	Message	Duration / Fetch
9	12:17:18	SELECT Population Max(Population) AS Max_Population FROM Country	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
10	12:17:56	SELECT MAX(Population) AS Max_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000
11	12:19:06	SELECT MAX(Population) AS Max_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000
12	12:19:06	SELECT MIN(Population) AS Min_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000
13	12:28:08	INSERT INTO Persons VALUES (215, 'Miriam', Null, 5, 105, 'Estonia'), (216, 'Nate', Null, 5, 108, 'Ireland')	Error Code: 1136. Column count doesn't match value count at row 1	0.000 sec
14	12:28:55	INSERT INTO Persons VALUES (215, 'Miriam', Null, 45000, 5, 105, 'Estonia'), (216, 'Nate', Null, 40000, 5, 108,...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
15	12:29:36	SELECT COUNT(Lname) AS Count_Lname FROM Persons LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000

12:29 PM
1/23/202

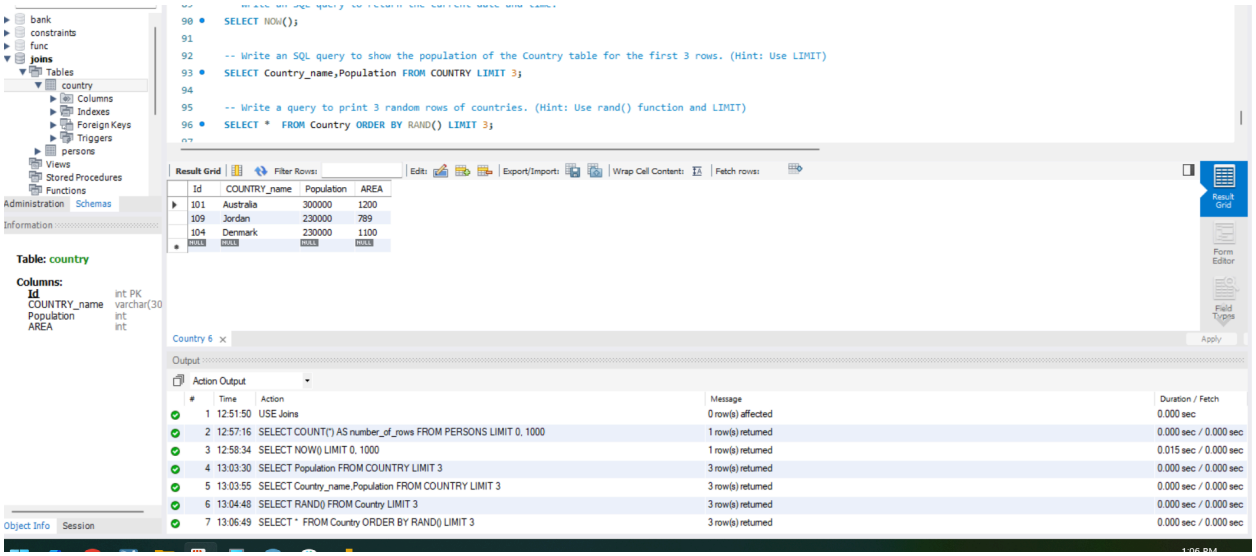
7. Write a query to find the number of rows in the Persons table.

8. Write an SQL query to return the current date and time.



9. Write an SQL query to show the population of the Country table for the first 3 rows. (Hint: Use LIMIT)

10. Write a query to print 3 random rows of countries. (Hint: Use rand() function and LIMIT)



Task 3:[User defined functions]

Questions:

Consider the Country table and Persons table that you created earlier and perform the following:

1. **Add a new column called DOB in Persons table with datatype as Date.**
2. **Write a user-defined function to calculate age using DOB.**
3. **Write a select query to fetch the Age of all persons reusing the function that has been created.**

```sql
-- user-defined functions
-- Add a new column called DOB in Persons table with datatype as Date.
ALTER TABLE Persons ADD COLUMN DOB Date;
UPDATE PERSONS SET DOB='1992-05-03' WHERE ID IN (201);
SELECT * FROM PERSONS;
UPDATE PERSONS SET DOB='1992-03-01' WHERE ID IN (202);
UPDATE PERSONS SET DOB='1989-05-03' WHERE ID IN (203);
UPDATE PERSONS SET DOB='1991-01-02' WHERE ID IN (204);
UPDATE PERSONS SET DOB='1985-06-09' WHERE ID IN (205);
UPDATE PERSONS SET DOB='1990-09-02' WHERE ID IN (206);
UPDATE PERSONS SET DOB='1989-04-03' WHERE ID IN (207);
UPDATE PERSONS SET DOB='1992-01-02' WHERE ID IN (208);
UPDATE PERSONS SET DOB='1993-02-08' WHERE ID IN (209);
UPDATE PERSONS SET DOB='1992-05-03' WHERE ID IN (210);

-- Write a user-defined function to calculate age using DOB.
DELIMITER $$
CREATE FUNCTION AGE(DOB Date)
RETURNS int
DETERMINISTIC
BEGIN
    DECLARE age INT;
    SET age=TIMESTAMPDIFF(Year, DOB, NOW());
    RETURN Age;
END $$
DELIMITER ;
```

```sql
DELIMITER ;
-- Write a select query to fetch the Age of all persons reusing the function that has been created.
SELECT Id, Fname, Lname,Country_name, DOB, Age(DOB) AS Current_age FROM Persons;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

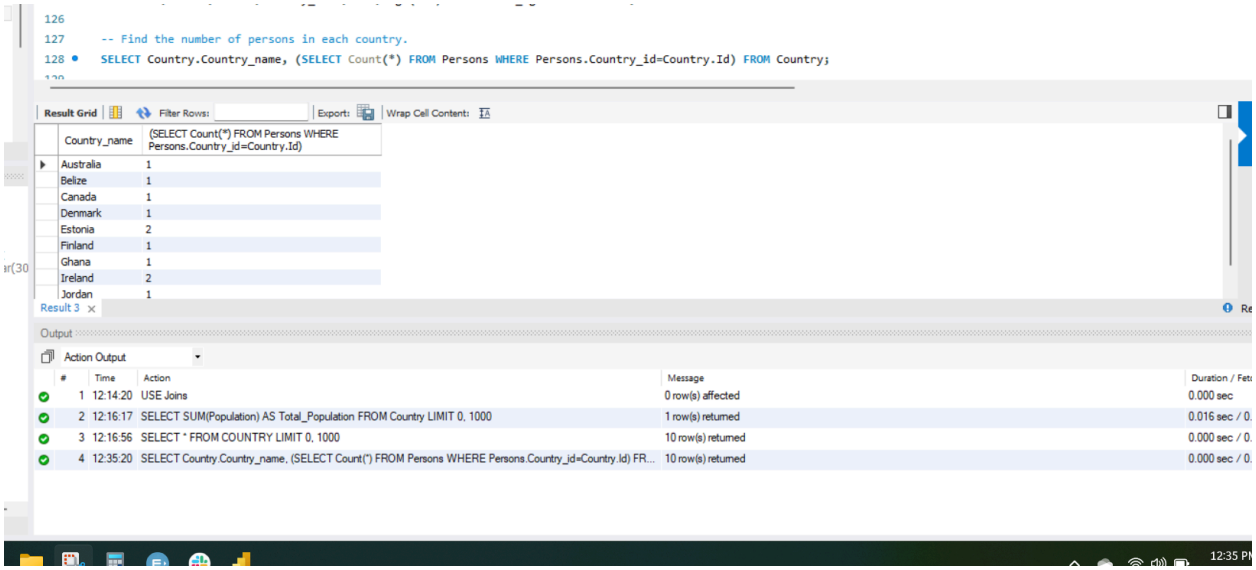
Id	Fname	Lname	Country_name	DOB	Current_age
201	John	Doe	Australia	1992-05-03	31
202	Jane	Smith	Belize	1992-03-01	31
203	Bob	Johnson	Canada	1989-05-03	34
204	Alice	Williams	Denmark	1991-01-02	33
205	Charlie	Brown	Estonia	1985-06-09	38
206	Eva	Anderson	Finland	1990-09-02	33
207	George	Miller	Ghana	1989-04-03	34
208	Ivy	Martin	Ireland	1992-01-02	32
209	Jack	Clark	Jordan	1993-02-08	30

Task 7:[Subqueries]

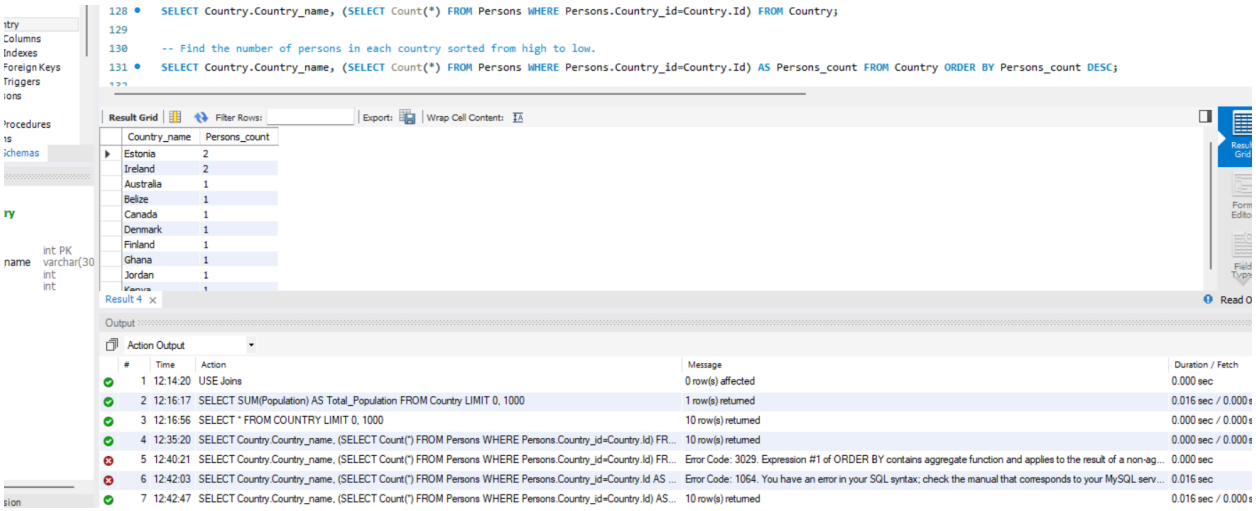
Questions:

Consider the Country table and Persons table that you created earlier and perform the following:

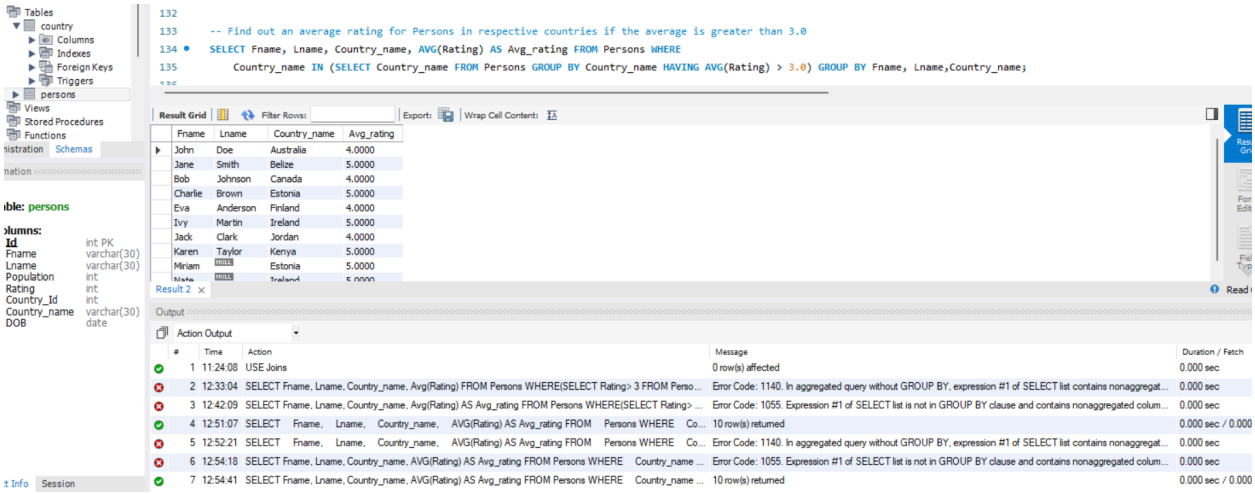
1. **Find the number of persons in each country.**



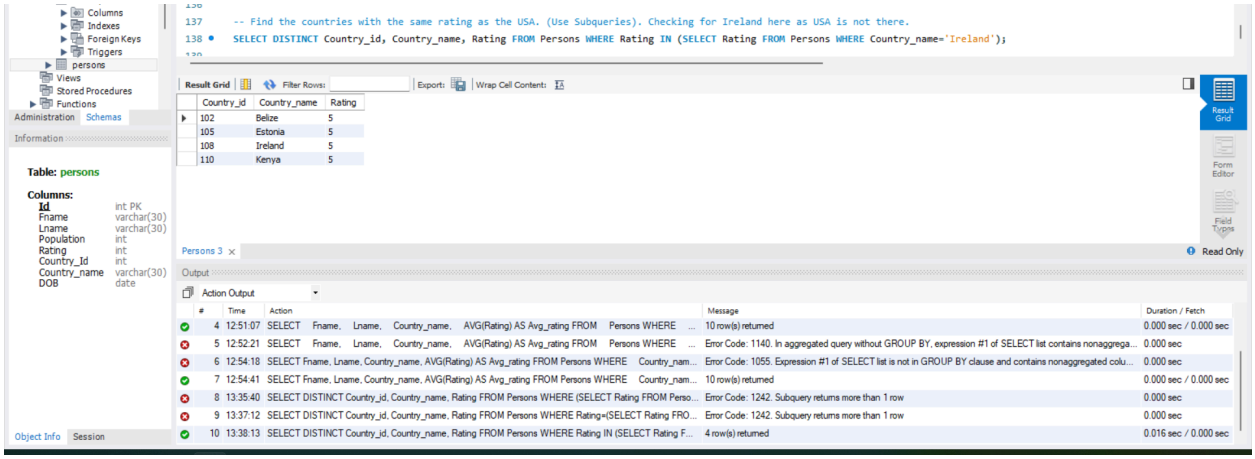
2. Find the number of persons in each country sorted from high to low.



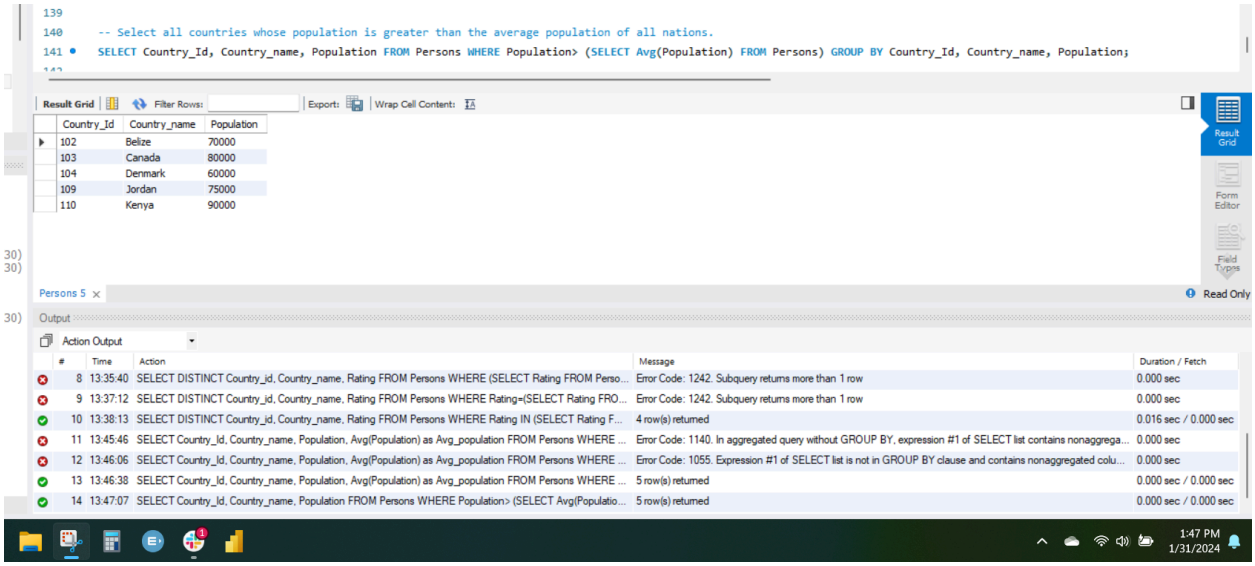
3. Find out an average rating for Persons in respective countries if the average is greater than 3.0



4. Find the countries with the same rating as the USA. (Use Subqueries)



5. Select all countries whose population is greater than the average population of all nations.

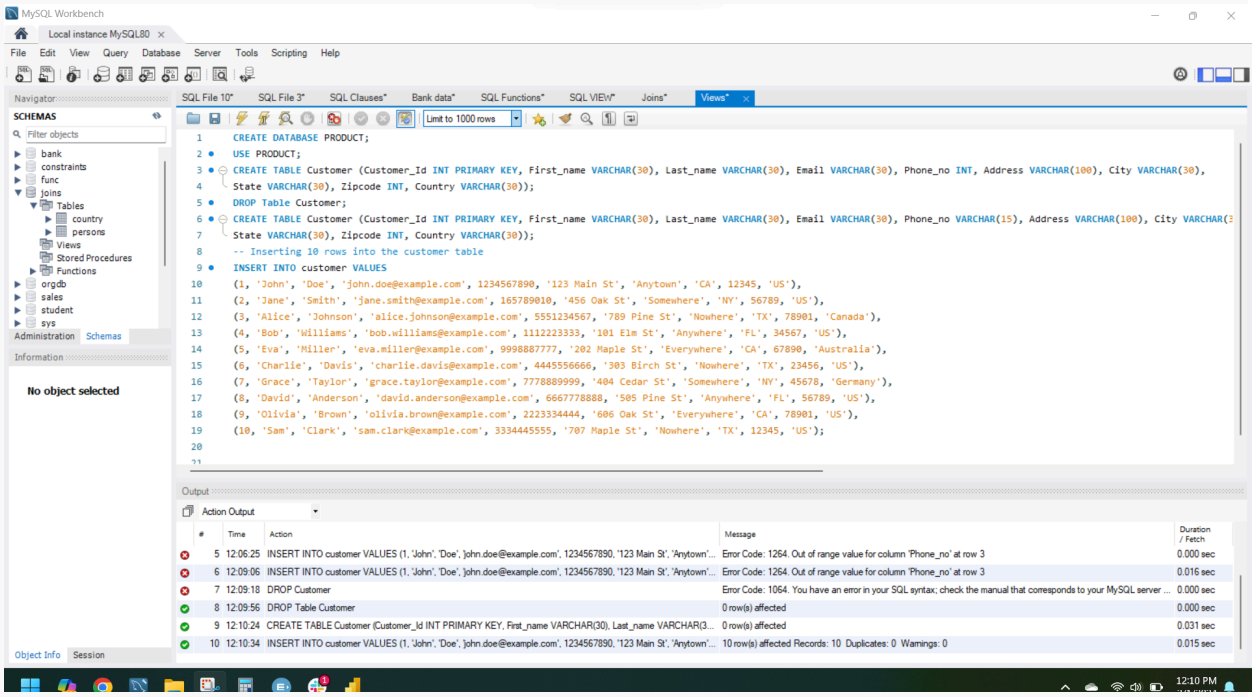


Task 8:[Views]

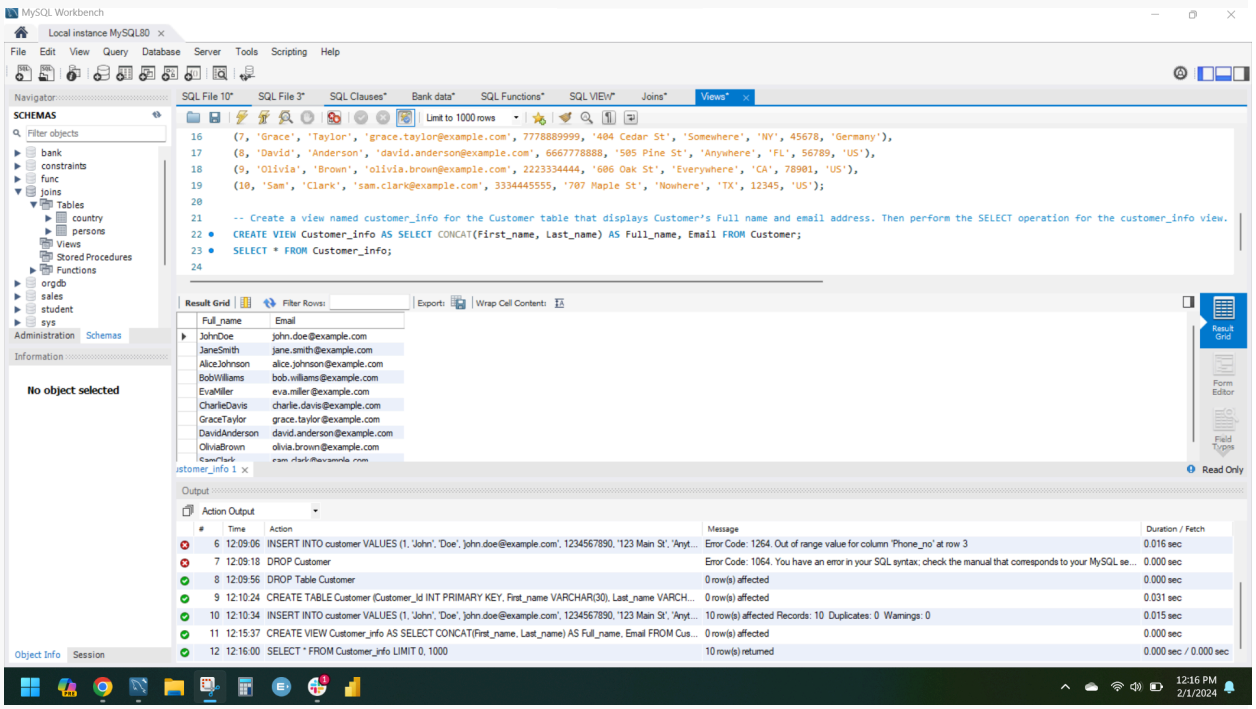
Questions:

Create a database named Product and create a table called Customer with the following fields in the Product database:

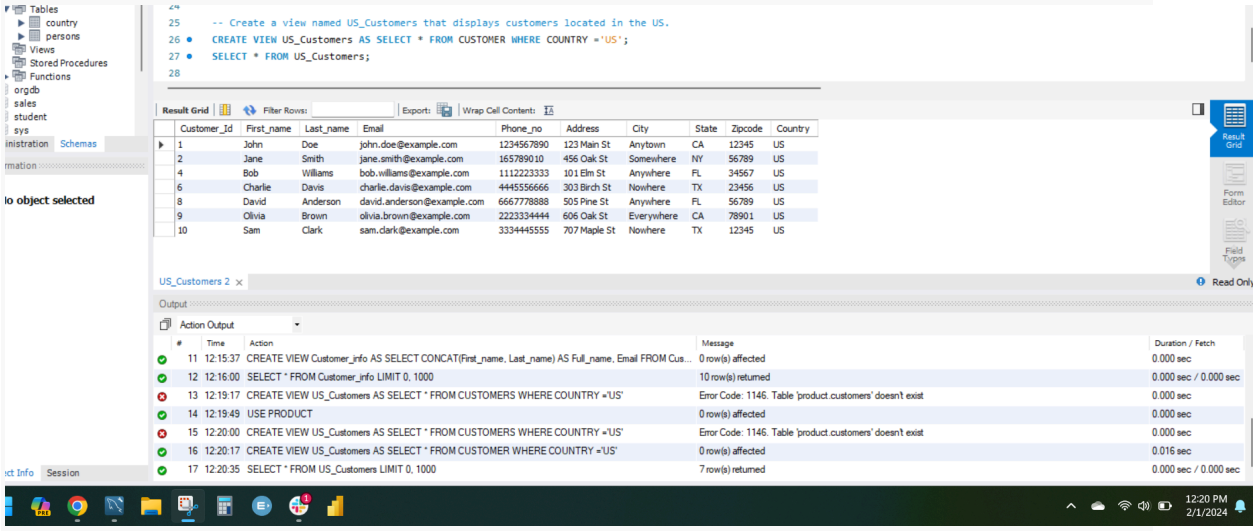
- **Customer_Id - Make PRIMARY KEY**
- **First_name**
- **Last_name**
- **Email**
- **Phone_no**
- **Address**
- **City**
- **State**
- **Zip_code**
- **Country**



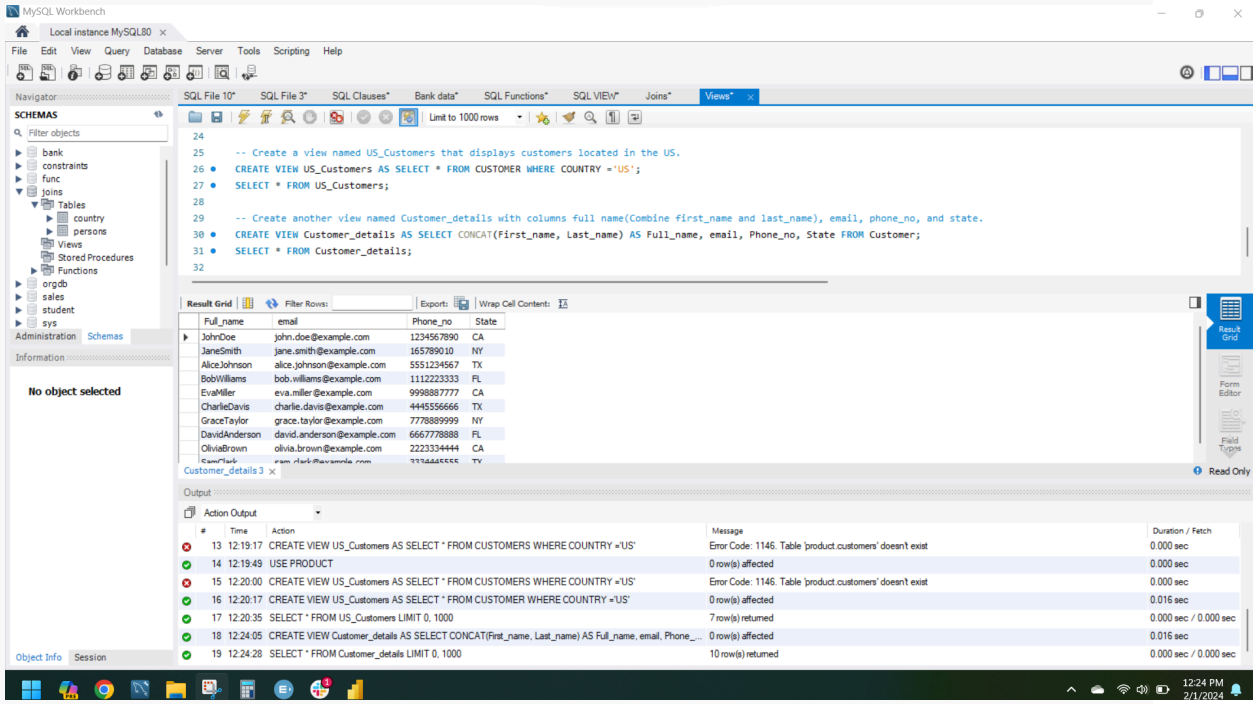
1. Create a view named customer_info for the Customer table that displays Customer's Full name and email address. Then perform the SELECT operation for the customer_info view.



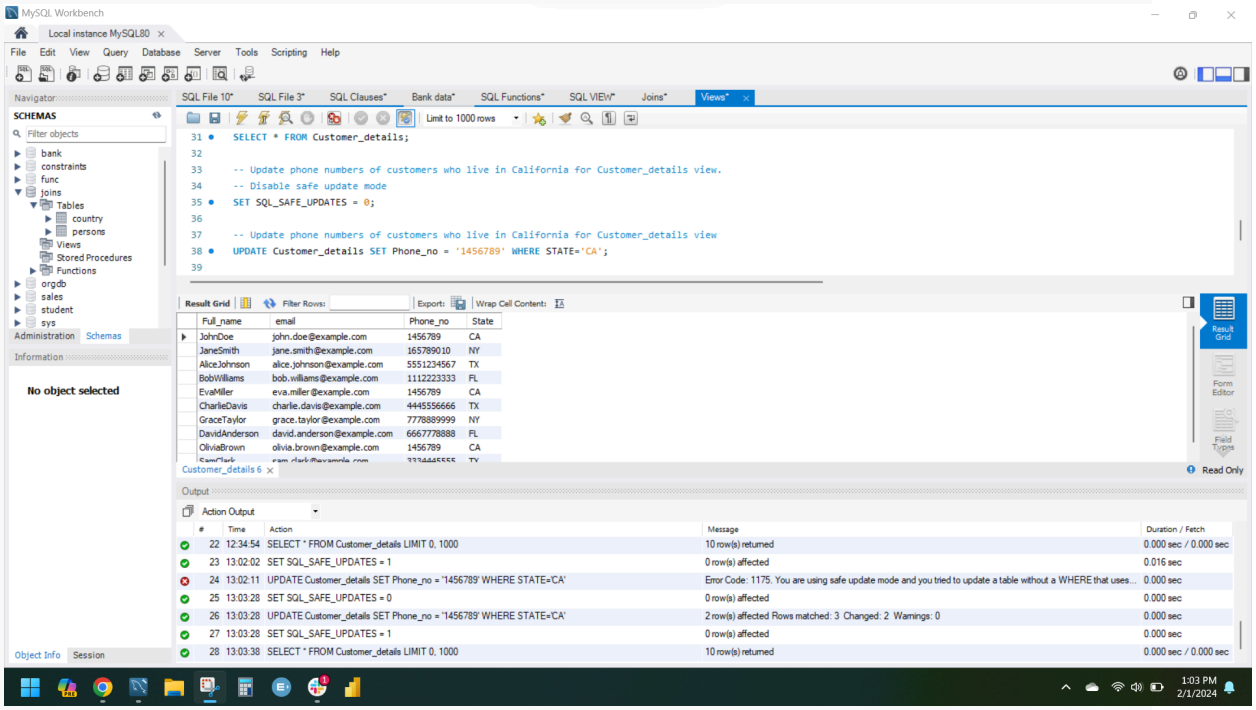
2. Create a view named US_Customers that displays customers located in the US.



3. Create another view named Customer_details with columns full name(Combine first_name and last_name), email, phone_no, and state.

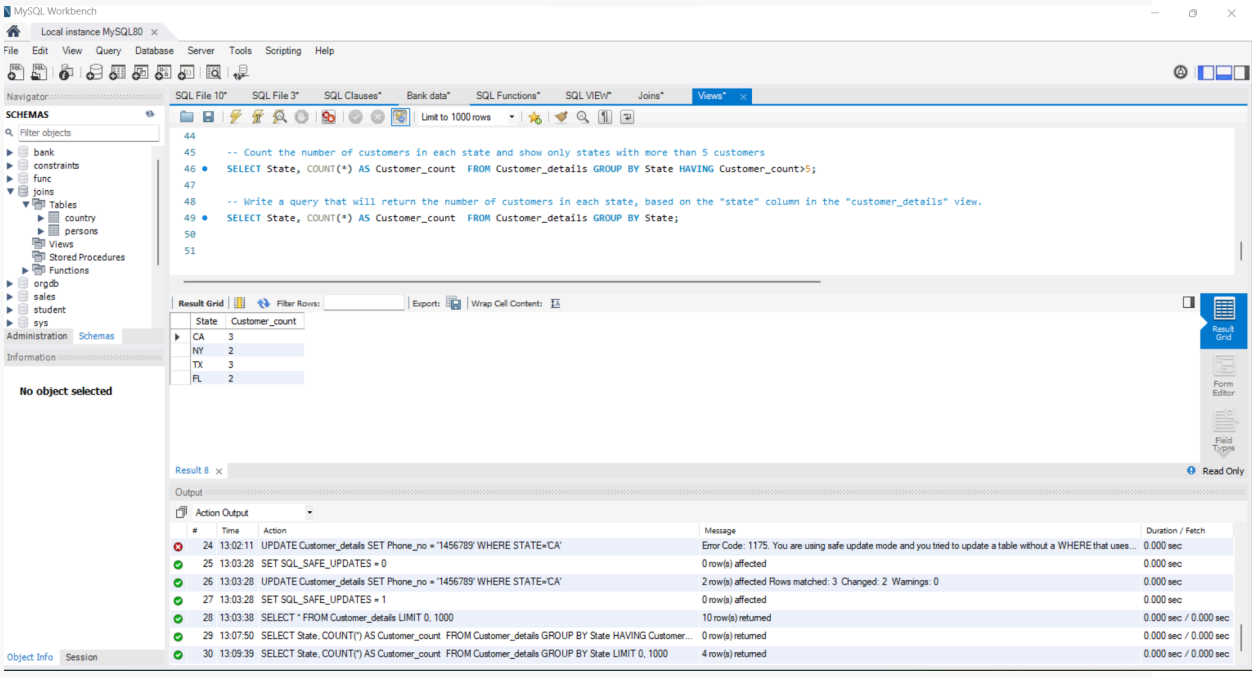


4. Update phone numbers of customers who live in California for Customer_details view.

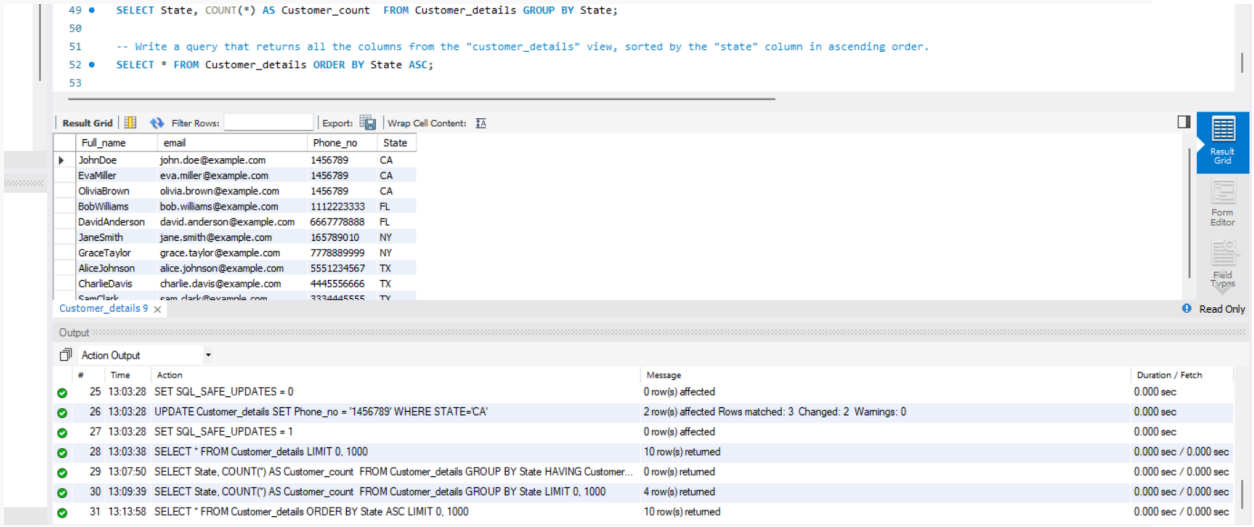


5. Count the number of customers in each state and show only states with more than 5 customers.

6. Write a query that will return the number of customers in each state, based on the "state" column in the "customer_details" view.



7.Write a query that returns all the columns from the "customer_details" view, sorted by the "state" column in ascending order.



```
49 •  SELECT State, COUNT(*) AS Customer_count  FROM Customer_details GROUP BY State;
50
51    -- Write a query that returns all the columns from the "customer_details" view, sorted by the "state" column in ascending order.
52 •  SELECT * FROM Customer_details ORDER BY State ASC;
53
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Full_name	email	Phone_no	State
JohnDoe	john.doe@example.com	1456789	CA
EvaMiller	eva.miller@example.com	1456789	CA
OliviaBrown	olivia.brown@example.com	1456789	CA
BobWilliams	bob.williams@example.com	1112223333	FL
DavidAnderson	david.anderson@example.com	6667778888	FL
JaneSmith	jane.smith@example.com	165789010	NY
GraceTaylor	grace.taylor@example.com	7778889999	NY
AliceJohnson	alice.johnson@example.com	5551234567	TX
CharlieDavis	charlie.davis@example.com	4445556666	TX
SamClark	sam.clark@example.com	3334445555	TX

Customer_details 9 | Read Only

Output

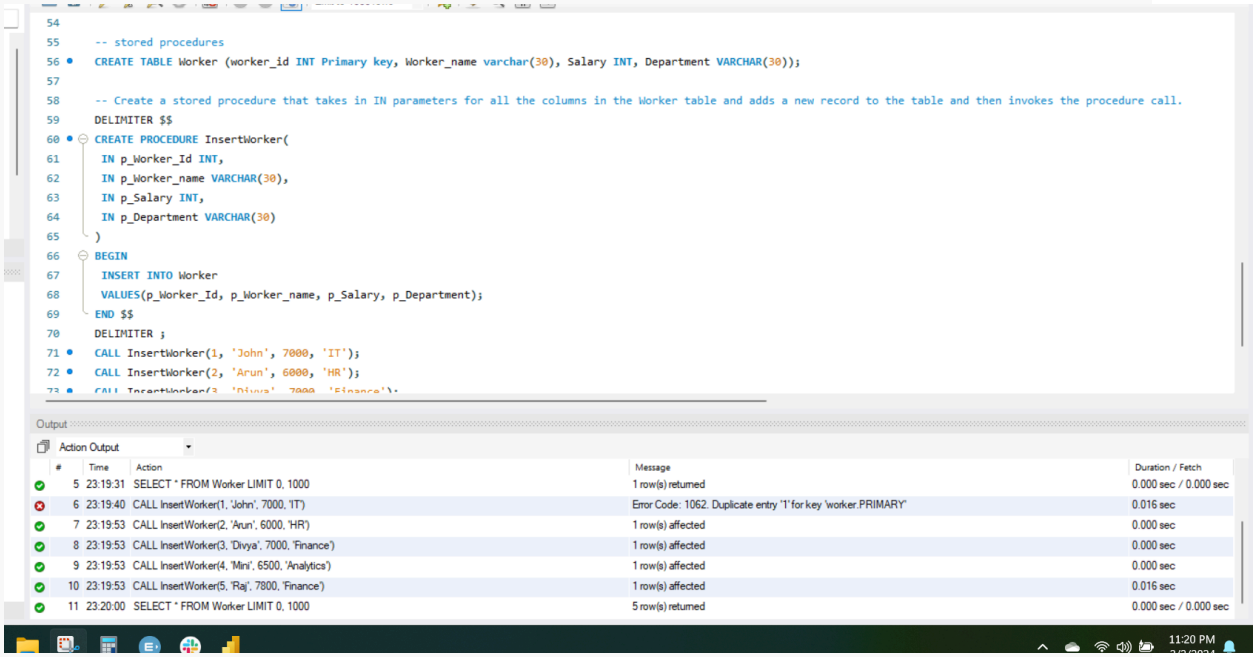
Action Output

#	Time	Action	Message	Duration / Fetch
25	13:03:28	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.000 sec
26	13:03:28	UPDATE Customer_details SET Phone_no = '1456789' WHERE STATE='CA'	2 row(s) affected Rows matched: 3 Changed: 2 Warnings: 0	0.000 sec
27	13:03:28	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec
28	13:03:38	SELECT * FROM Customer_details LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
29	13:07:50	SELECT State, COUNT(*) AS Customer_count FROM Customer_details GROUP BY State HAVING Customer...	0 row(s) returned	0.000 sec / 0.000 sec
30	13:09:39	SELECT State, COUNT(*) AS Customer_count FROM Customer_details GROUP BY State LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
31	13:13:58	SELECT * FROM Customer_details ORDER BY State ASC LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Task 9:[Stored Procedures]

Questions:

Create a stored procedure that takes in IN parameters for all the columns in the Worker table and adds a new record to the table and then invokes the procedure call.



```
54
55    -- stored procedures
56 •  CREATE TABLE Worker (worker_id INT Primary key, Worker_name varchar(30), Salary INT, Department VARCHAR(30));
57
58    -- Create a stored procedure that takes in IN parameters for all the columns in the Worker table and adds a new record to the table and then invokes the procedure call.
59    DELIMITER $$
60 •  CREATE PROCEDURE InsertWorker(
61      IN p_Worker_Id INT,
62      IN p_Worker_name VARCHAR(30),
63      IN p_Salary INT,
64      IN p_Department VARCHAR(30)
65    )
66    BEGIN
67      INSERT INTO Worker
68      VALUES(p_Worker_Id, p_Worker_name, p_Salary, p_Department);
69    END $$
70    DELIMITER ;
71 •  CALL InsertWorker(1, 'John', 7000, 'IT');
72 •  CALL InsertWorker(2, 'Arun', 6000, 'HR');
73 •  CALL InsertWorker(3, 'Divya', 7000, 'Finance');
```

Output

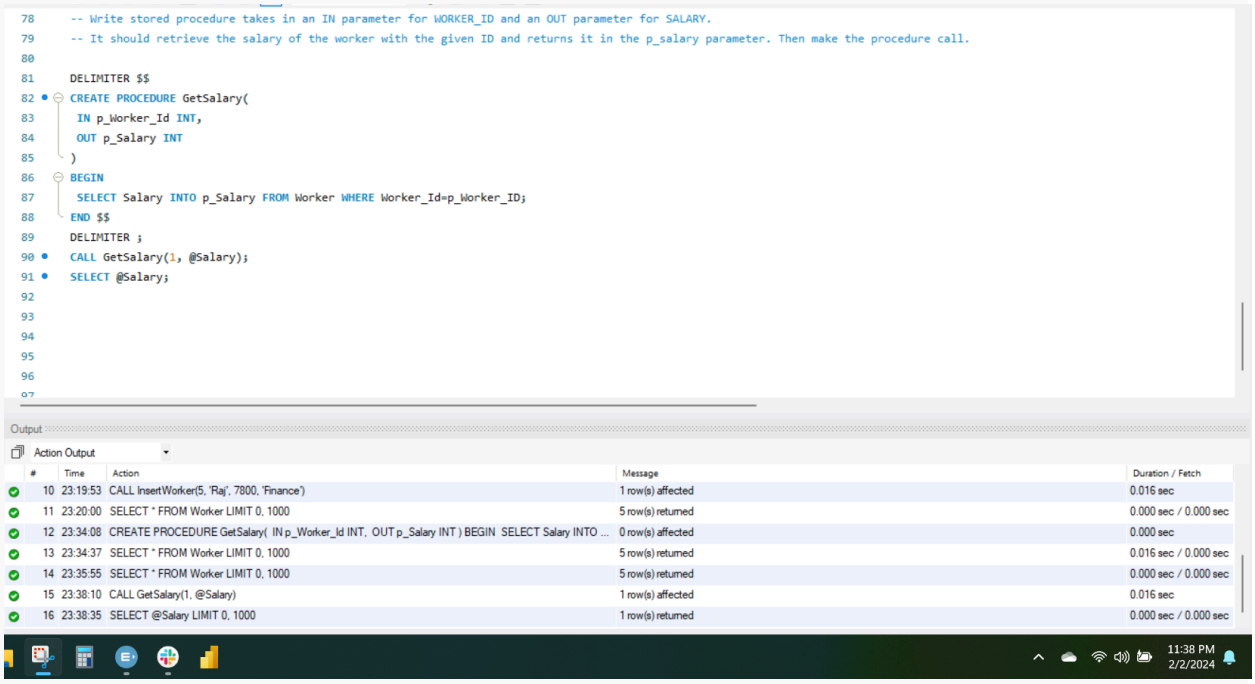
Action Output

#	Time	Action	Message	Duration / Fetch
5	23:19:31	SELECT * FROM Worker LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
6	23:19:40	CALL InsertWorker(1, 'John', 7000, 'IT')	Error Code: 1062. Duplicate entry '1' for key 'worker.PRIMARY'	0.016 sec
7	23:19:53	CALL InsertWorker(2, 'Arun', 6000, 'HR')	1 row(s) affected	0.000 sec
8	23:19:53	CALL InsertWorker(3, 'Divya', 7000, 'Finance')	1 row(s) affected	0.000 sec
9	23:19:53	CALL InsertWorker(4, 'Mini', 6500, 'Analytics')	1 row(s) affected	0.000 sec
10	23:19:53	CALL InsertWorker(5, 'Raj', 7800, 'Finance')	1 row(s) affected	0.016 sec
11	23:20:00	SELECT * FROM Worker LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

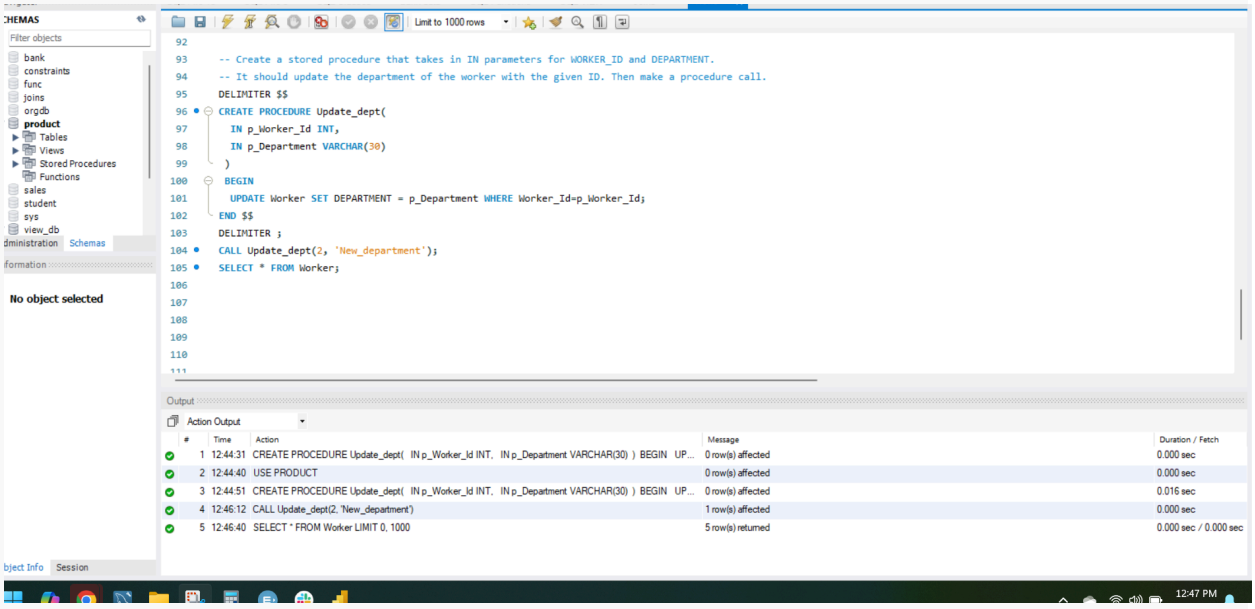
11:20 PM

Write stored procedure takes in an IN parameter for WORKER_ID and an OUT parameter for SALARY. It should retrieve the salary of the worker with the

given ID and returns it in the p_salary parameter. Then make the procedure call.

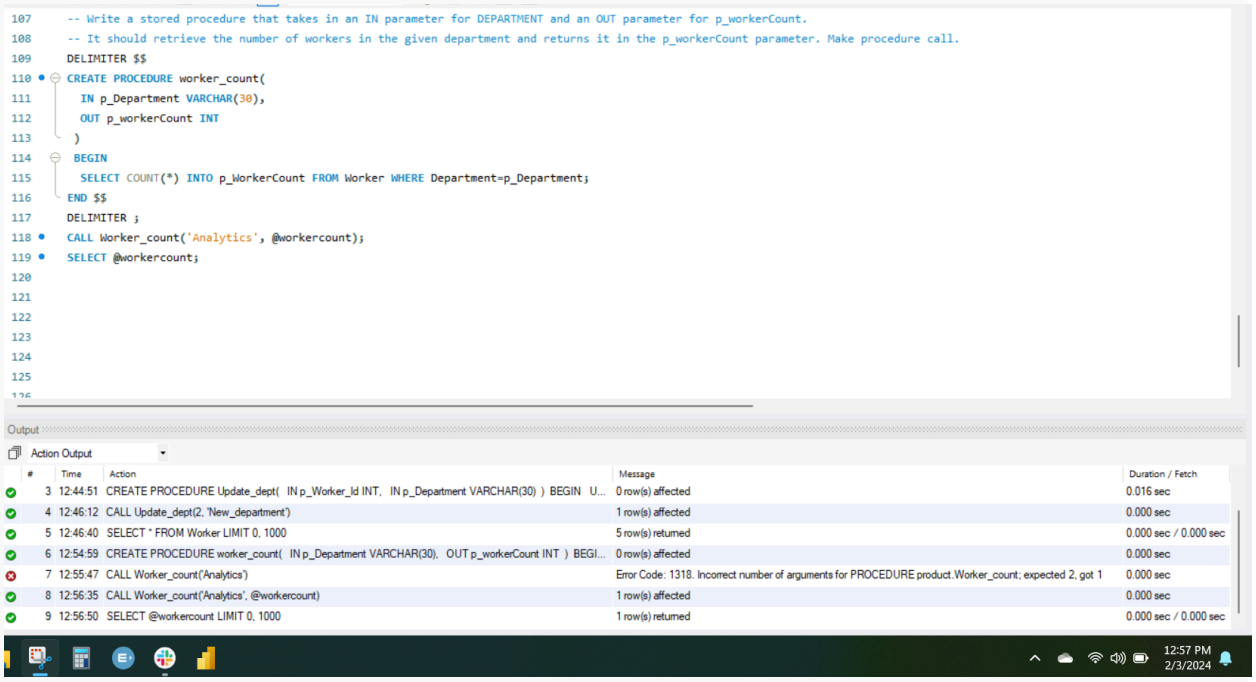


3. Create a stored procedure that takes in IN parameters for WORKER_ID and DEPARTMENT. It should update the department of the worker with the given ID. Then make a procedure call.



4.Write a stored procedure that takes in an IN parameter for DEPARTMENT and

an OUT parameter for p_workerCount. It should retrieve the number of workers in the given department and returns it in the p_workerCount parameter. Make procedure call.



```
-- Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_workerCount.
-- It should retrieve the number of workers in the given department and returns it in the p_workerCount parameter. Make procedure call.
DELIMITER $$
CREATE PROCEDURE worker_count(
  IN p_Department VARCHAR(30),
  OUT p_workerCount INT
 )
 BEGIN
  SELECT COUNT(*) INTO p_WorkerCount FROM Worker WHERE Department=p_Department;
 END $$
DELIMITER ;
CALL Worker_count('Analytics', @workercount);
SELECT @workercount;
```

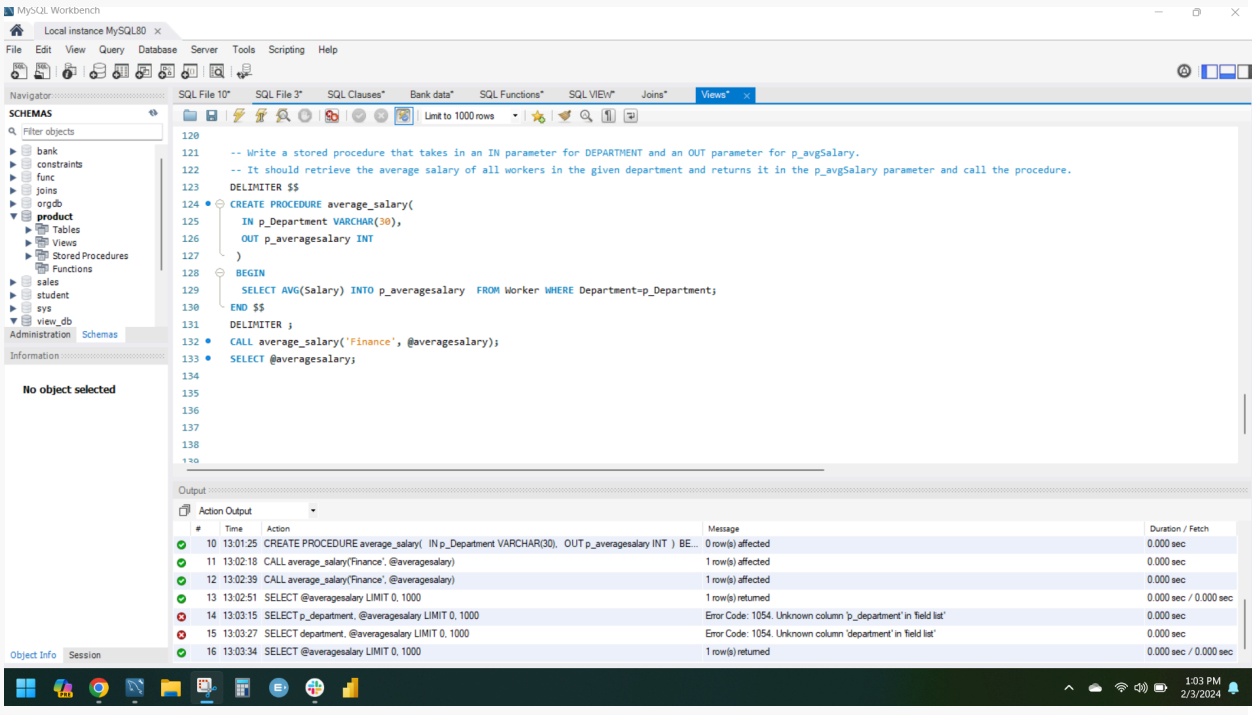
Output

Action Output

#	Time	Action	Message	Duration / Fetch
3	12:44:51	CREATE PROCEDURE Update_dept(IN p_Worker_Id INT, IN p_Department VARCHAR(30)) BEGIN U...	0 row(s) affected	0.016 sec
4	12:46:12	CALL Update_dept(2, 'New_department')	1 row(s) affected	0.000 sec
5	12:46:40	SELECT * FROM Worker LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
6	12:54:59	CREATE PROCEDURE worker_count(IN p_Department VARCHAR(30), OUT p_workerCount INT) BEGI...	0 row(s) affected	0.000 sec
7	12:55:47	CALL Worker_count('Analytics')	Error Code: 1318. Incorrect number of arguments for PROCEDURE product.Worker_count; expected 2, got 1	0.000 sec
8	12:56:35	CALL Worker_count('Analytics', @workercount)	1 row(s) affected	0.000 sec
9	12:56:50	SELECT @workercount LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

12:57 PM 2/3/2024

5. Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_avgSalary. It should retrieve the average salary of all workers in the given department and returns it in the p_avgSalary parameter and call the procedure.



```
-- Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_avgSalary.
-- It should retrieve the average salary of all workers in the given department and returns it in the p_avgSalary parameter and call the procedure.
DELIMITER $$
CREATE PROCEDURE average_salary(
  IN p_Department VARCHAR(30),
  OUT p_averagesalary INT
 )
 BEGIN
  SELECT AVG(Salary) INTO p_averagesalary  FROM Worker WHERE Department=p_Department;
 END $$
DELIMITER ;
CALL average_salary('Finance', @averagesalary);
SELECT @averagesalary;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
10	13:01:25	CREATE PROCEDURE average_salary(IN p_Department VARCHAR(30), OUT p_averagesalary INT) BE...	0 row(s) affected	0.000 sec
11	13:02:18	CALL average_salary('Finance', @averagesalary)	1 row(s) affected	0.000 sec
12	13:02:39	CALL average_salary('Finance', @averagesalary)	1 row(s) affected	0.000 sec
13	13:02:51	SELECT @averagesalary LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
14	13:03:15	SELECT p_department, @averagesalary LIMIT 0, 1000	Error Code: 1054. Unknown column 'p_department' in 'field list'	0.000 sec
15	13:03:27	SELECT department, @averagesalary LIMIT 0, 1000	Error Code: 1054. Unknown column 'department' in 'field list'	0.000 sec
16	13:03:34	SELECT @averagesalary LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

1:03 PM 2/3/2024

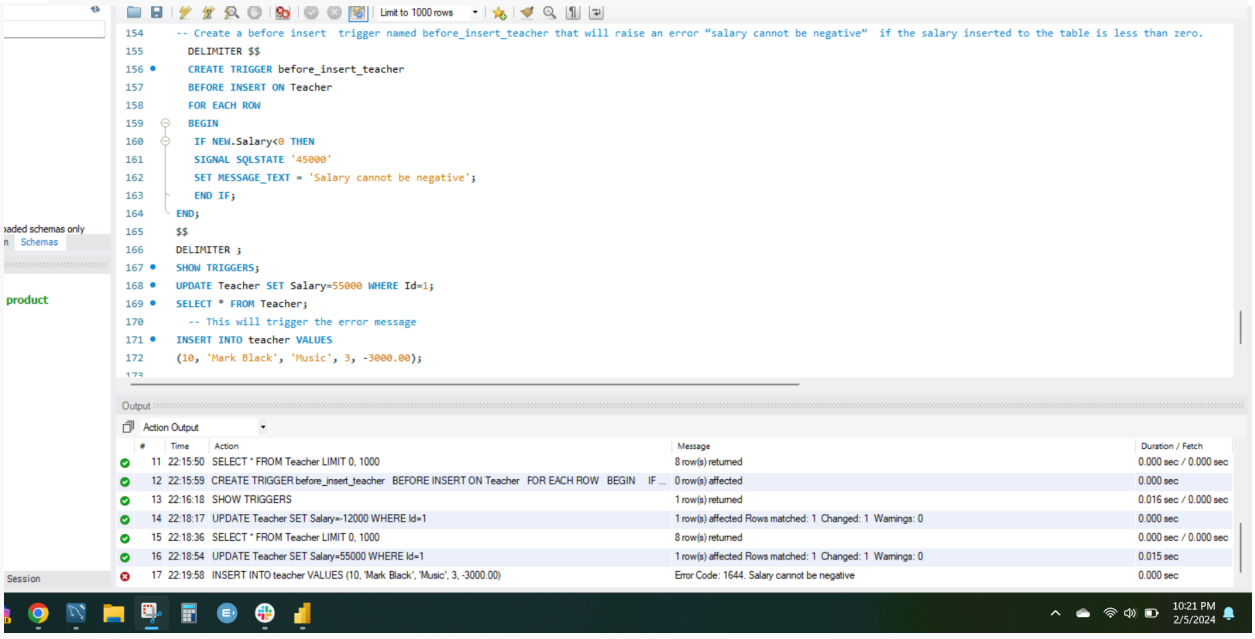
Task 10:[Triggers]

Questions:

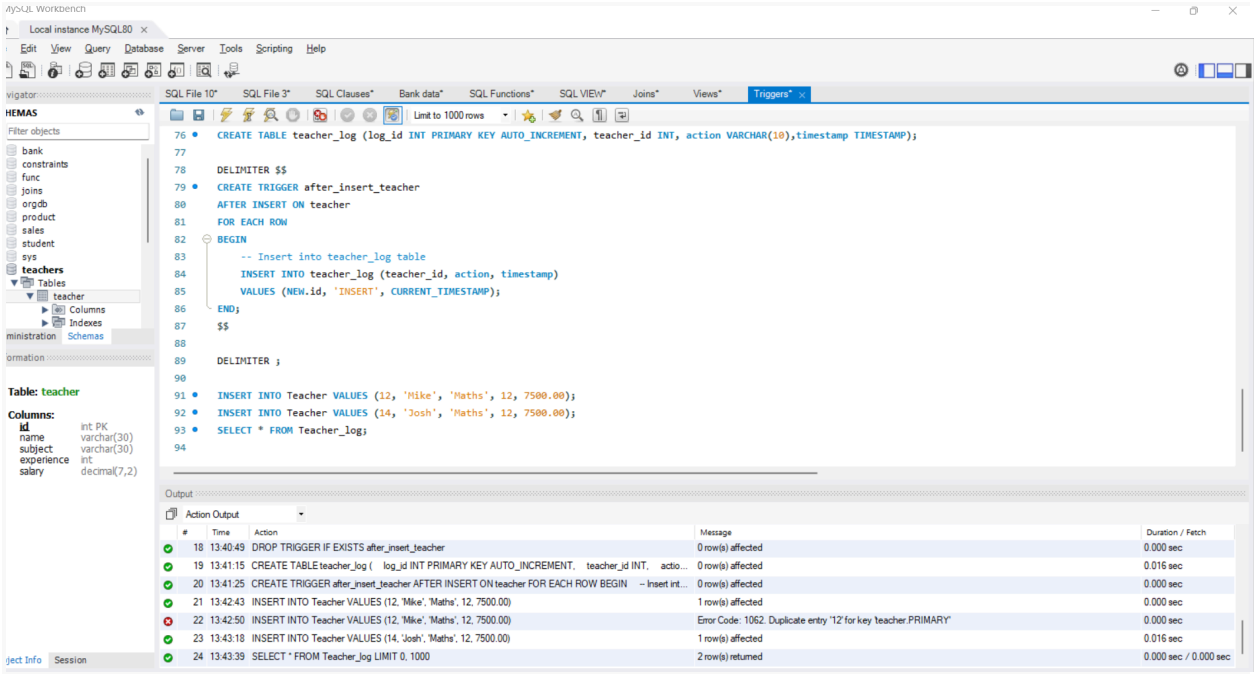
1. **Create a table named teachers with fields id, name, subject, experience and salary and insert 8 rows.**

```
-- Triggers
-- Create a table named teachers with fields id, name, subject, experience and salary and insert 8 rows.
CREATE TABLE Teacher(Id INT Primary key, Name VARCHAR(30), Subject VARCHAR(30), Experience INT, SALARY DECIMAL(7,2));
INSERT INTO Teacher VALUES
(1, 'John Miller', 'Math', 5, 5000.00),
(2, 'Jane Smith', 'English', 8, 60000.00),
(3, 'Bob Johnson', 'Science', 3, 45000.00),
(4, 'Alice Brown', 'History', 10, 70000.00),
(5, 'David Wilson', 'Computer Science', 6, 55000.00),
(6, 'Sara Miller', 'Physics', 4, 48000.00),
(7, 'Mike Davis', 'Chemistry', 7, 62000.00),
(8, 'Emily White', 'Biology', 2, 42000.00);
```

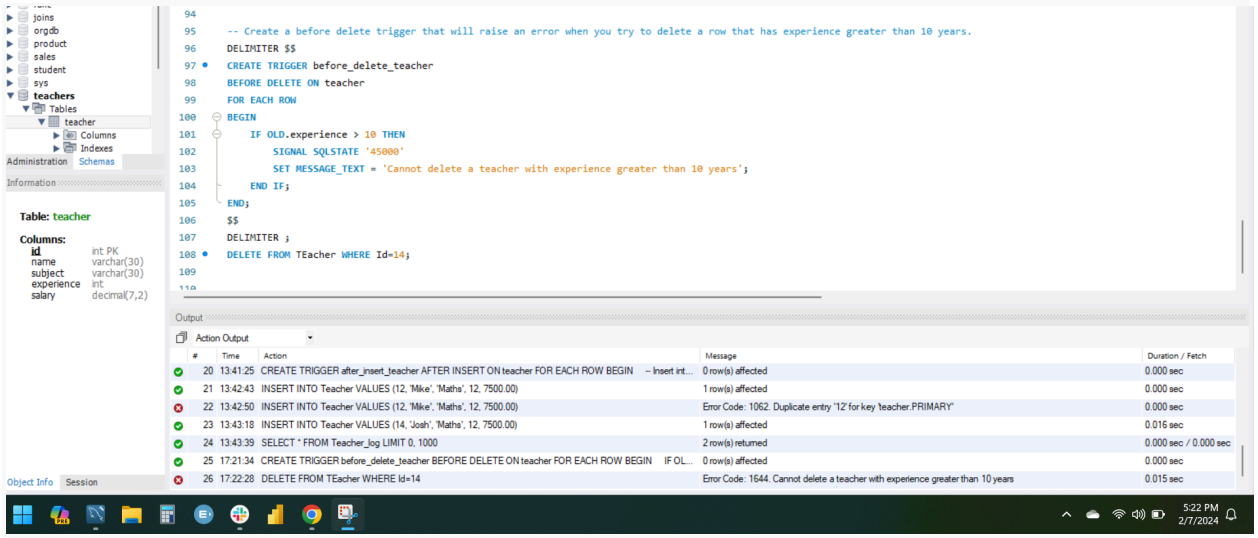
2. **Create a before insert trigger named before_insert_teacher that will raise an error "salary cannot be negative" if the salary inserted to the table is less than zero.**



3. **Create an after insert trigger named after_insert_teacher that inserts a row with teacher_id,action, timestamp to a table called teacher_log when a new gets inserted to the teacher table. tecaher_id -> column of teacher table, action -> the trigger action, timestamp -> time at which the new row has got inserted.**



4. **Create a before delete trigger that will raise an error when you try to delete a row that has experience greater than 10 years.**



5. **Create an after delete trigger that will insert a row to teacher_log table when that row is deleted from teacher table.**

```
110
111     -- Create an after delete trigger that will insert a row to teacher_log table when that row is deleted from teacher table.
112     DELIMITER $$
113     CREATE TRIGGER after_delete_teacher
114     AFTER DELETE ON teacher
115     FOR EACH ROW
116     BEGIN
117         INSERT INTO teacher_log (teacher_id, action, timestamp)
118         VALUES (OLD.id, 'DELETE', CURRENT_TIMESTAMP);
119     END;
120     $$
121
122     DELIMITER ;
123     DELETE FROM Teacher WHERE Id=13;
124     SELECT * FROM Teacher;
125     SELECT * FROM Teacher_log;
126
127     SHOW TRIGGERS;
128
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
29	17:26:04	DELETE FROM Teacher WHERE Id=9	0 row(s) affected	0.000 sec
30	17:26:22	SELECT * FROM Teacher_log LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
31	17:27:56	SHOW TRIGGERS	3 row(s) returned	0.000 sec / 0.000 sec
32	17:28:34	SELECT * FROM Teacher LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
33	17:30:58	INSERT INTO Teacher VALUES (13, 'Mike', 'Maths', 1, 7500.00)	1 row(s) affected	0.000 sec
34	17:31:10	DELETE FROM Teacher WHERE Id=13	1 row(s) affected	0.000 sec
35	17:31:15	SELECT * FROM Teacher_log LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec