

SUBQUERIES AND VIEWS

➤ DISPLAYING TABLES **COUNTRY** AND **PERSONS**

A screenshot of the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view of database objects. The 'Country' table is selected under the 'person' schema. The main editor window displays the SQL query:

```
1 # USING COUNTRY TABLE AND PERSON TABLE
2
3 SELECT * FROM Country;
4 SELECT * FROM Persons;
```

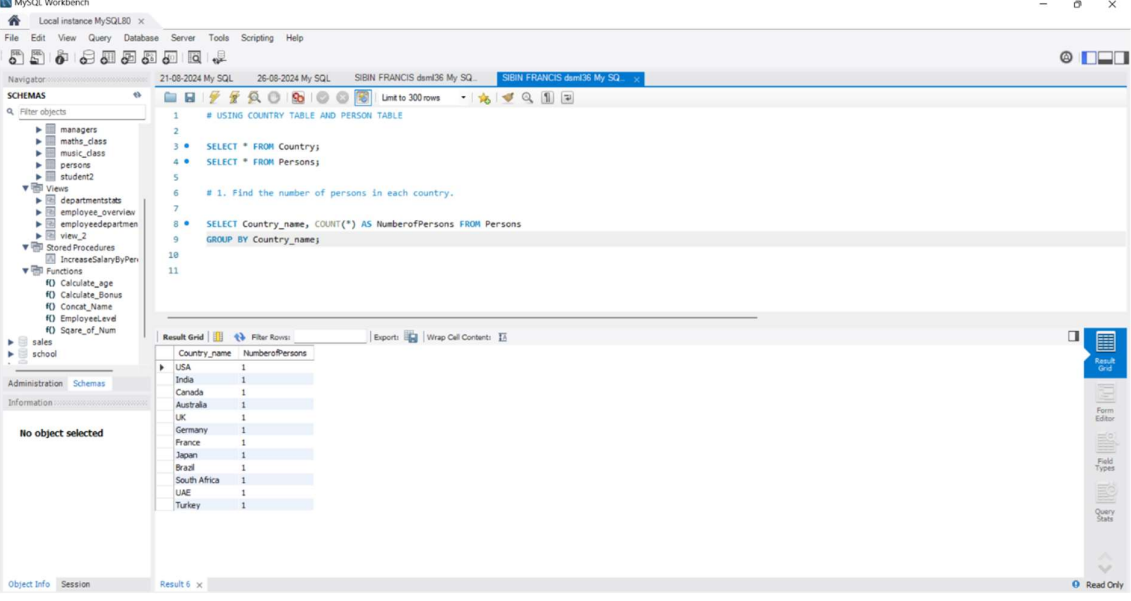
 The 'Result Grid' at the bottom shows the data for the 'Country' table with columns: Id, Country_name, Population, and Area. The data is as follows:

Id	Country_name	Population	Area
1	USA	331000000	California
2	India	1380000000	Maharashtra
3	Canada	38000000	Ontario
4	Australia	25600000	New South Wales
5	UK	67000000	London
6	Germany	83000000	Bavaria
7	France	67000000	Île-de-France
8	Japan	125000000	Tokyo
9	Brazil	213000000	São Paulo
10	South Africa	60000000	Gauteng
11	UAE	33150000	Abu Dhabi
12	Turkey	138500000	Istanbul
13	Latvia	30150000	Vilnius
14	Russia	128500000	Moscow

A screenshot of the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view of database objects. The 'Persons' table is selected under the 'person' schema. The main editor window displays the same SQL query as the previous screenshot. The 'Result Grid' at the bottom shows the data for the 'Persons' table with columns: Id, Fname, Lname, Population, Rating, Country_Id, Country_name, and DOB. The data is as follows:

Id	Fname	Lname	Population	Rating	Country_Id	Country_name	DOB
1	John	Doe	1000000	4.5	1	USA	1990-05-15
2	Jane	Smith	2000000	4.8	2	India	1985-12-22
3	Michael	Brown	1500000	4.2	3	Canada	2000-03-10
4	Emily	Davis	2500000	3.9	4	Australia	1992-07-25
5	James	Wilson	1800000	4	5	UK	1988-11-30
6	Anna	Moore	1300000	4.6	6	Germany	1975-06-18
7	Robert	Taylor	1700000	3.7	7	France	1999-09-14
8	Linda	Anderson	1400000	4.9	8	Japan	2001-01-01
9	David	Thomas	2100000	3.8	9	Brazil	1983-04-12
10	Sarah	Jackson	1600000	4.3	10	South Africa	1980-08-20
11	Janice		1200000	4.6	11	UAE	2005-03-21
12	Richie		2200000	4.9	12	Turkey	1996-02-29

➤ FINDING NUMBER OF PERSONS IN EACH COUNTRY



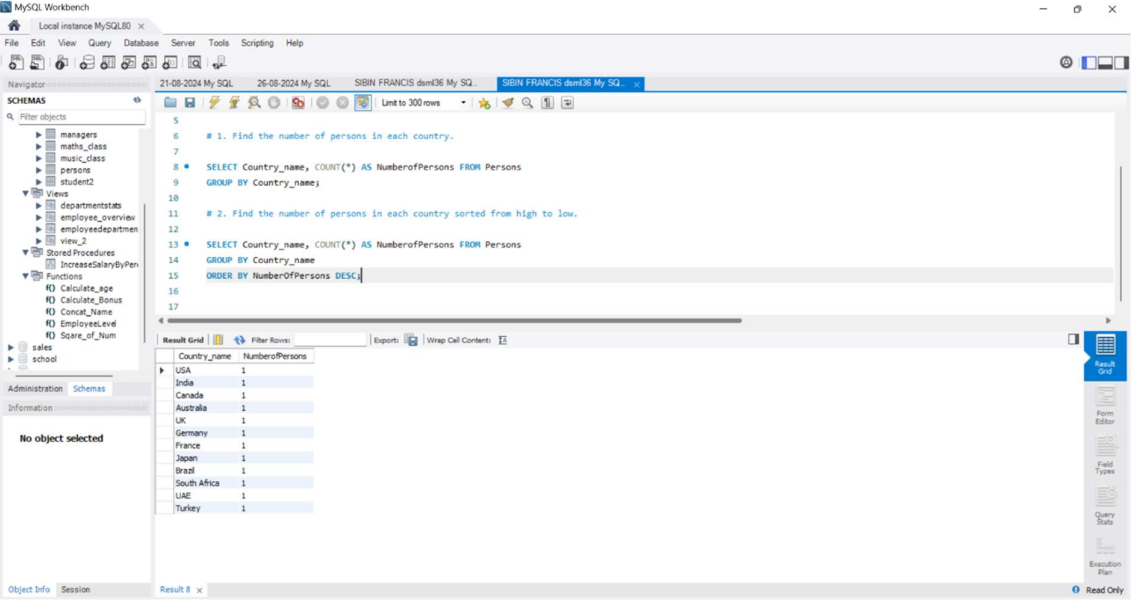
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with various databases like 'managers', 'maths_class', 'music_class', 'persons', 'student2', 'departmentstates', 'employee_overview', 'employee_department', 'view_2', 'Stored Procedures', 'Functions', 'sales', and 'school'. The main editor window contains a SQL query:

```
1 # USING COUNTRY TABLE AND PERSON TABLE
2
3 SELECT * FROM Country;
4 SELECT * FROM Persons;
5
6 # 1. Find the number of persons in each country.
7
8 SELECT Country_name, COUNT(*) AS NumberOfPersons FROM Persons
9 GROUP BY Country_name;
```

The 'Result Grid' at the bottom shows the output of the query:

Country_name	NumberOfPersons
USA	1
India	1
Canada	1
Australia	1
UK	1
Germany	1
France	1
Japan	1
Brazil	1
South Africa	1
UAE	1
Turkey	1

➤ FINDING NUMBER OF PERSONS IN EACH COUNTRY SORTED FROM HIGH TO LOW.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree. The main editor window contains a SQL query:

```
5
6 # 1. Find the number of persons in each country.
7
8 SELECT Country_name, COUNT(*) AS NumberOfPersons FROM Persons
9 GROUP BY Country_name;
10
11 # 2. Find the number of persons in each country sorted from high to low.
12
13 SELECT Country_name, COUNT(*) AS NumberOfPersons FROM Persons
14 GROUP BY Country_name
15 ORDER BY NumberOfPersons DESC;
```

The 'Result Grid' at the bottom shows the output of the query, sorted from high to low:

Country_name	NumberOfPersons
USA	1
India	1
Canada	1
Australia	1
UK	1
Germany	1
France	1
Japan	1
Brazil	1
South Africa	1
UAE	1
Turkey	1

➤ AVERAGE RATING FOR PERSONS IN RESPECTIVE COUNTRIES IF THE AVERAGE > THAN 3.0

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
9 GROUP BY Country_name);
10
11 # 2. Find the number of persons in each country sorted from high to low.
12
13 • SELECT Country_name, COUNT(*) AS NumberOfPersons FROM Persons
14 GROUP BY Country_name
15 ORDER BY NumberOfPersons DESC;
16
17 # 3. Find out an average rating for Persons in respective countries if the average is greater than 3.0
18
19 • SELECT Country_name, AVG(Rating) AS AverageRating FROM Persons
20 GROUP BY Country_name
21 HAVING AverageRating > 3.0 ;
```

The result grid shows the following data:

Country_name	AverageRating
USA	4.5
India	4.800000190734863
Canada	4.199999801265127
Australia	3.9000000953674316
UK	4
Germany	4.599999904632568
France	3.700000047683716
Japan	4.900000015367432
Brazil	3.799999952116284
South Africa	4.300000190734863
UAE	4.599999904632568
Turkey	4.900000095367432

➤ COUNTRIES WITH THE SAME RATING AS THE USA.

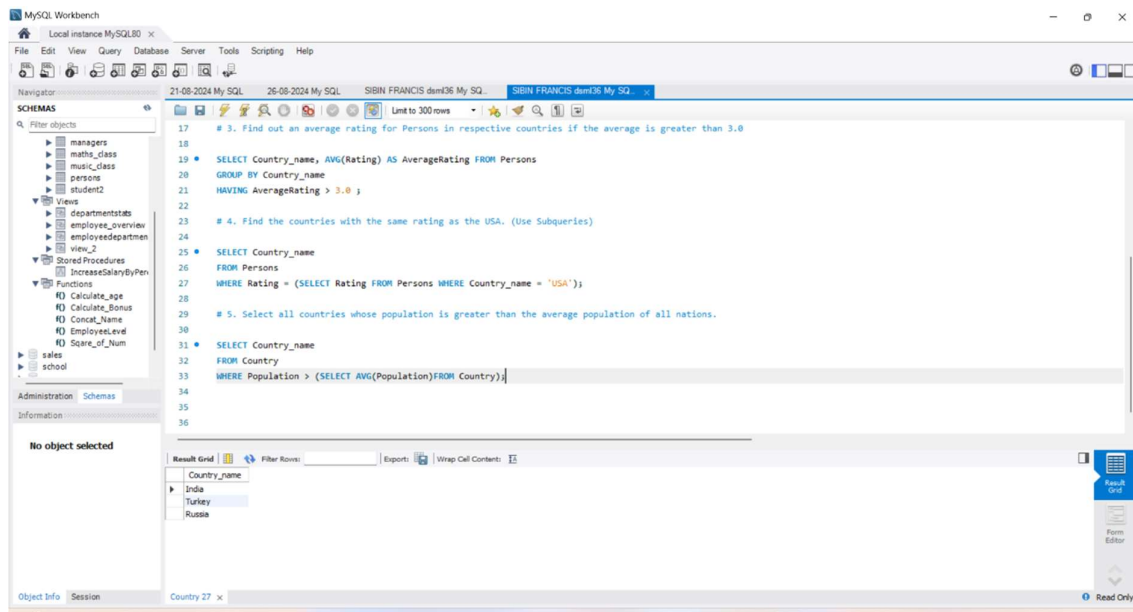
The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
15 ORDER BY NumberOfPersons DESC;
16
17 # 3. Find out an average rating for Persons in respective countries if the average is greater than 3.0
18
19 • SELECT Country_name, AVG(Rating) AS AverageRating FROM Persons
20 GROUP BY Country_name
21 HAVING AverageRating > 3.0 ;
22
23 # 4. Find the countries with the same rating as the USA. (Use Subqueries)
24
25 • SELECT Country_name
26 FROM Persons
27 WHERE Rating = (SELECT Rating FROM Persons WHERE Country_name = 'USA');
28
```

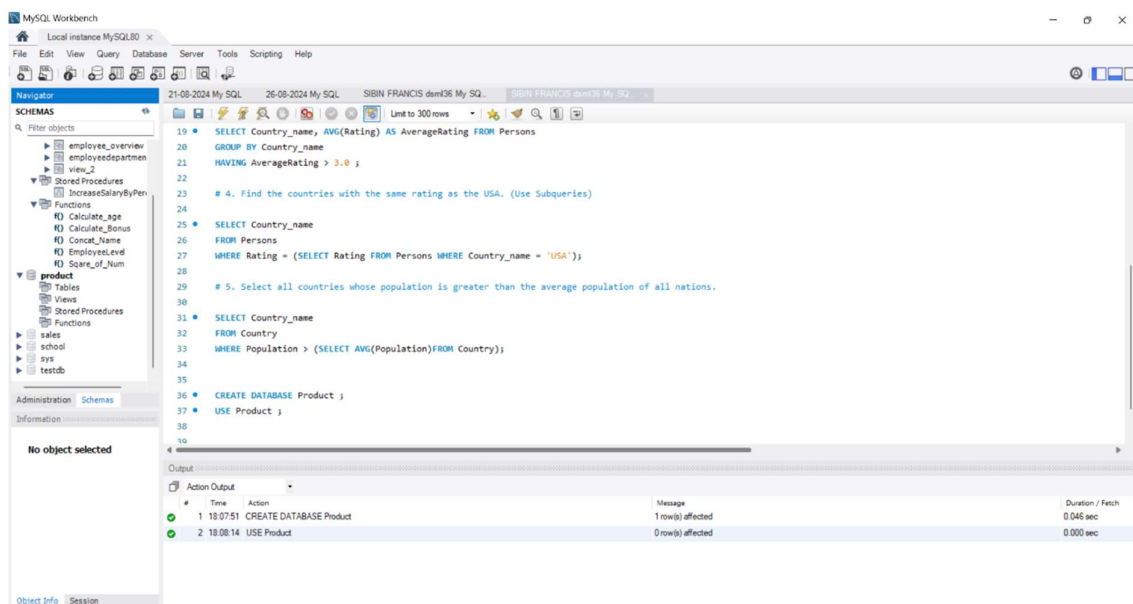
The result grid shows the following data:

Country_name
USA

➤ COUNTRIES WHOSE POPULATION > THAN THE AVERAGE POPULATION OF ALL NATIONS.



➤ CREATED DATA BASE PRODUCT



➤ INSERTED VALUES FOR PRODUCT

The screenshot shows the MySQL Workbench interface with the following SQL queries:

```
28
29 # 5. Select all countries whose population is greater than the average population of all nations.
30
31 • SELECT Country_name
32 FROM Country
33 WHERE Population > (SELECT AVG(Population)FROM Country);
34
35 # VIEWS
36
37 • CREATE DATABASE Product ;
38 • USE Product ;
39
40 • CREATE TABLE Customer(
41 Customer_Id INT PRIMARY KEY,
42 First_name VARCHAR(25),
43 Last_name VARCHAR(25),
44 Email VARCHAR(100),
45 Phone_no VARCHAR (15),
46 Address VARCHAR (100),
47 City VARCHAR (50),
48 State VARCHAR (10),
49 Zip_code VARCHAR(10),
50 Country VARCHAR (50)
51 )
52
53 • INSERT INTO Customer (Customer_Id, First_name, Last_name, Email, Phone_no, Address, City, State, Zip_code, Country)
54 VALUES
55 (1, 'John', 'Doe', 'john.doe@example.com', '+1-2025550908', '123 Elm St', 'Los Angeles', 'California', '90001', 'USA'),
56 (2, 'Jane', 'Smith', 'jane.smith@example.com', '+1-2026650188', '456 Maple Ave', 'New York', 'New York', '10001', 'USA'),
57 (3, 'Michael', 'Johnson', 'm.johnson@example.com', '+1-2025550188', '789 Oak Blvd', 'Chicago', 'Illinois', '60601', 'USA'),
```

The screenshot shows the MySQL Workbench interface with the following SQL queries and the resulting data in the Customer table:

```
51 )
52
53 • INSERT INTO Customer (Customer_Id, First_name, Last_name, Email, Phone_no, Address, City, State, Zip_code, Country)
54 VALUES
55 (1, 'John', 'Doe', 'john.doe@example.com', '+1-2025550908', '123 Elm St', 'Los Angeles', 'California', '90001', 'USA'),
56 (2, 'Jane', 'Smith', 'jane.smith@example.com', '+1-2026650188', '456 Maple Ave', 'New York', 'New York', '10001', 'USA'),
57 (3, 'Michael', 'Johnson', 'm.johnson@example.com', '+1-2025550188', '789 Oak Blvd', 'Chicago', 'Illinois', '60601', 'USA'),
58 (4, 'Aisha', 'Patel', 'aisha.patel@example.com', '+61-412345678', '321 George Street', 'Sydney', 'New South Wales', '2000', 'Australia'),
59 (5, 'Vikram', 'Singh', 'vikram.singh@example.com', '+91-15209876543', '654 Hauptstrasse', 'Berlin', 'Berlin', '10115', 'Germany'),
60 (6, 'Sanya', 'Reddy', 'sanya.reddy@example.com', '+33-612345678', '987 Rue de Rivoli', 'Paris', 'Île-de-France', '75001', 'France'),
61 (7, 'Rohit', 'Bose', 'rohit.bose@example.com', '+81-9080765432', '123 Shibuya Crossing', 'Tokyo', 'Tokyo', '150-0001', 'Japan'),
62 (8, 'Sneha', 'Gupta', 'sneha.gupta@example.com', '+27-723456789', '456 Nelson Mandela Blvd', 'Cape Town', 'Western Cape', '7800', 'South Africa'),
63 (9, 'Manish', 'Verma', 'manish.verma@example.com', '+55-11987654321', '789 Avenida Paulista', 'São Paulo', 'São Paulo', '01318-100', 'Brazil'),
64 (10, 'Anjali', 'Desai', 'anjali.desai@example.com', '+91-9988776655', '321 Banjara Hills', 'Hyderabad', 'Telangana', '500034', 'India');
65
66 • SELECT * FROM Customer;
```

Customer_Id	First_name	Last_name	Email	Phone_no	Address	City	State	Zip_code	Country
1	John	Doe	john.doe@example.com	+1-2025550908	123 Elm St	Los Angeles	California	90001	USA
2	Jane	Smith	jane.smith@example.com	+1-2026650188	456 Maple Ave	New York	New York	10001	USA
3	Michael	Johnson	m.johnson@example.com	+1-2025550188	789 Oak Blvd	Chicago	Illinois	60601	USA
4	Aisha	Patel	aisha.patel@example.com	+61-412345678	321 George Street	Sydney	New South Wales	2000	Australia
5	Vikram	Singh	vikram.singh@example.com	+91-15209876543	654 Hauptstrasse	Berlin	Berlin	10115	Germany
6	Sanya	Reddy	sanya.reddy@example.com	+33-612345678	987 Rue de Rivoli	Paris	Île-de-France	75001	France
7	Rohit	Bose	rohit.bose@example.com	+81-9080765432	123 Shibuya Crossing	Tokyo	Tokyo	150-0001	Japan
8	Sneha	Gupta	sneha.gupta@example.com	+27-723456789	456 Nelson Mandela Blvd	Cape Town	Western Cape	7800	South Africa
9	Manish	Verma	manish.verma@example.com	+55-11987654321	789 Avenida Paulista	São Paulo	São Paulo	01318-100	Brazil
10	Anjali	Desai	anjali.desai@example.com	+91-9988776655	321 Banjara Hills	Hyderabad	Telangana	500034	India

➤ CREATED A VIEW NAMED **CUSTOMER_INFO** FOR THE CUSTOMER TABLE

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
60 (6, 'Sanya', 'Reddy', 'sanya.reddy@example.com', '+33-612345678', '987 Rue de Rivoli', 'Paris', 'Île-de-France', '75001', 'France'),
61 (7, 'Rohit', 'Bose', 'rohit.bose@example.com', '+81-9098765432', '123 Shibuya Crossing', 'Tokyo', 'Tokyo', '150-0001', 'Japan'),
62 (8, 'Sneha', 'Gupta', 'sneha.gupta@example.com', '+27-723456789', '456 Nelson Mandela Blvd', 'Cape Town', 'Western Cape', '8001', 'South Africa'),
63 (9, 'Manish', 'Verma', 'manish.verma@example.com', '+55-11987654321', '789 Avenida Paulista', 'São Paulo', 'São Paulo', '01310-100', 'Brazil'),
64 (10, 'Anjali', 'Desai', 'anjali.desai@example.com', '+91-9988776655', '321 Banjara Hills', 'Hyderabad', 'Telangana', '500034', 'India');
65
66 * SELECT * FROM Customer;
67
68 # 1. Create a view named customer_info for the Customer table that displays Customer's Full name and email address.
69 # Then perform the SELECT operation for the customer_info view.
70
71 * CREATE VIEW customer_info AS
72 SELECT CONCAT(First_name, ' ', Last_name) AS Full_name, Email
73 FROM Customer;
74
75 * SELECT * FROM customer_info;
76
```

The Results grid shows the output of the SELECT statement:

Full_name	Email
John Doe	john.doe@example.com
Jane Smith	jane.smith@example.com
Michael Johnson	m.johnson@example.com
Aisha Patel	aisha.patel@example.com
Vikram Singh	vikram.singh@example.com
Sanya Reddy	sanya.reddy@example.com
Rohit Bose	rohit.bose@example.com
Sneha Gupta	sneha.gupta@example.com
Manish Verma	manish.verma@example.com
Anjali Desai	anjali.desai@example.com

➤ CREATED A VIEW NAMED **US_CUSTOMERS** THAT DISPLAYS CUSTOMERS LOCATED IN THE US.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
71 * CREATE VIEW customer_info AS
72 SELECT CONCAT(First_name, ' ', Last_name) AS Full_name, Email
73 FROM Customer;
74
75 * SELECT * FROM customer_info;
76
77 # 2. Create a view named US_Customers that displays customers located in the US.
78
79 * CREATE VIEW US_Customers AS
80 SELECT * FROM Customer
81 WHERE Country = 'USA';
82
83 * SELECT * FROM US_Customers;
84
```

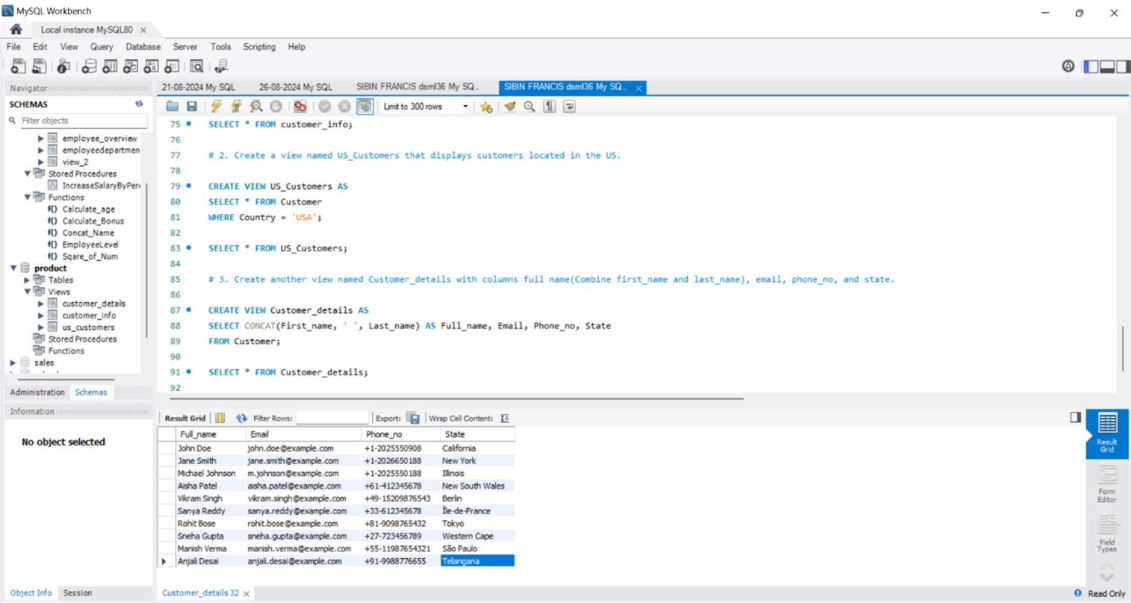
The Results grid shows the output of the SELECT statement:

Customer_Id	First_name	Last_name	Email	Phone_no	Address	City	State	Zip_code	Country
1	John	Doe	john.doe@example.com	+1-202550908	123 Elm St	Los Angeles	California	90001	USA
2	Jane	Smith	jane.smith@example.com	+1-2026650188	456 Maple Ave	New York	New York	10001	USA
3	Michael	Johnson	m.johnson@example.com	+1-2023530188	789 Oak Blvd	Chicago	Illinois	60601	USA

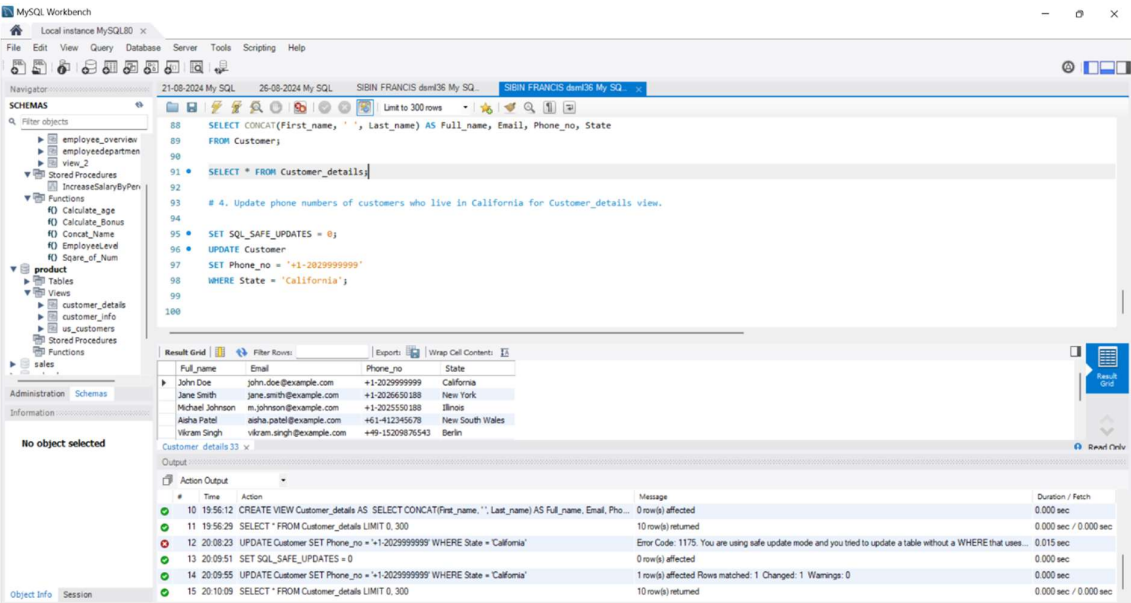
The Action Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	18:47:35	INSERT INTO Customer (Customer_Id, First_name, Last_name, Email, Phone_no, Address, City, State, Zip_code, Country)	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.032 sec
5	18:49:57	SELECT * FROM Customer LIMIT 0, 300	10 row(s) returned	0.000 sec / 0.000 sec
6	19:45:37	CREATE VIEW customer_info AS SELECT CONCAT(First_name, ' ', Last_name) AS Full_name, Email FROM Customer	10 row(s) affected	0.016 sec
7	19:45:53	SELECT * FROM customer_info LIMIT 0, 300	10 row(s) returned	0.000 sec / 0.000 sec
8	19:52:47	CREATE VIEW US_Customers AS SELECT * FROM Customer WHERE Country = 'USA'	10 row(s) affected	0.000 sec
9	19:53:12	SELECT * FROM US_Customers LIMIT 0, 300	3 row(s) returned	0.000 sec / 0.000 sec

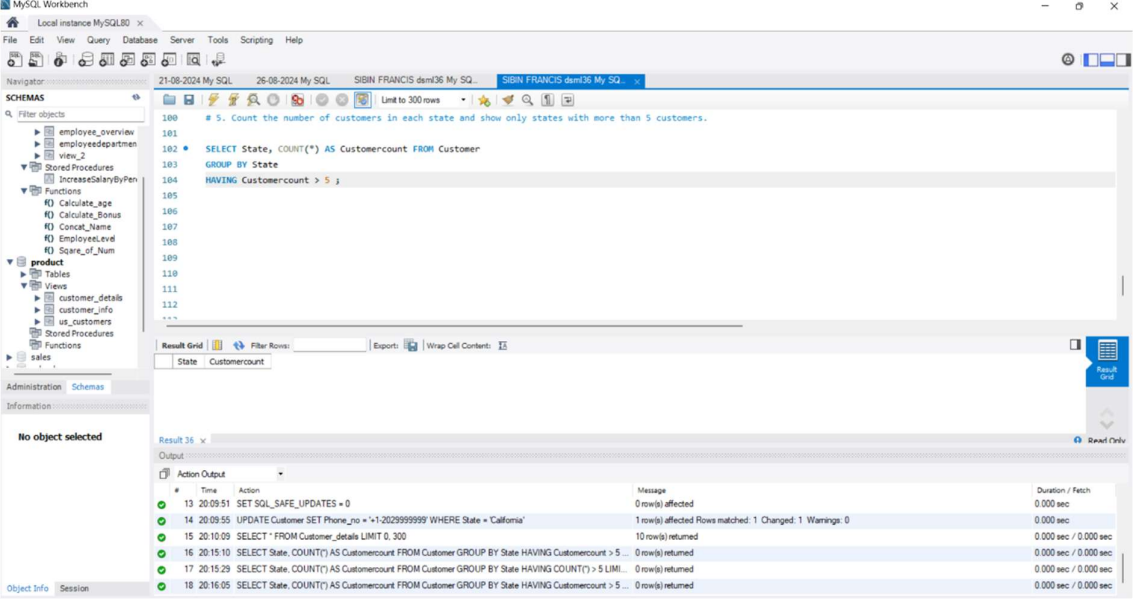
CREATED ANOTHER VIEW NAMED
CUSTOMER_DETAILS WITH COLUMNS FULL NAME,
EMAIL, PHONE_NO, AND STATE.



➤ UPDATED PHONE NUMBERS OF CUSTOMERS
WHO LIVE IN CALIFORNIA



➤ COUNTED THE NUMBER OF CUSTOMERS IN EACH STATE AND SHOWN ONLY STATES WITH MORE THAN 5 CUSTOMERS



The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

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

SELECT State, COUNT(*) AS Customercount FROM Customer
GROUP BY State
HAVING Customercount > 5 ;
```

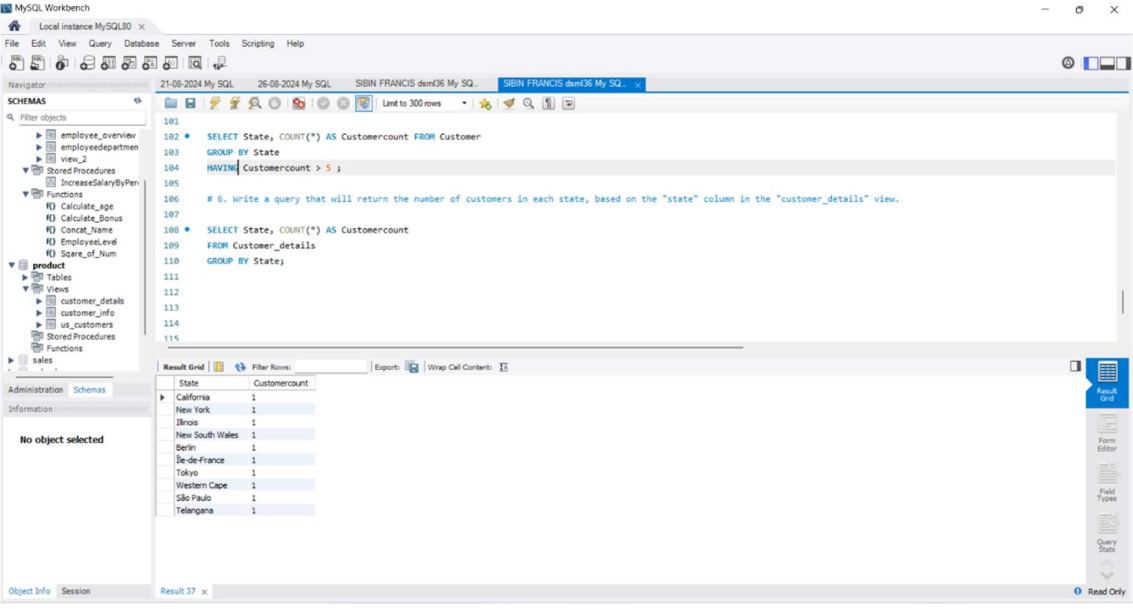
The Results grid shows the output of the query:

State	Customercount
California	1
New York	1
Illinois	1
New South Wales	1
Berlin	1
Île-de-France	1
Tokyo	1
Western Cape	1
São Paulo	1
Telangana	1

The Action Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
13	20:09:51	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.000 sec
14	20:09:55	UPDATE Customer SET Phone_no = '+1-2029999999' WHERE State = 'California'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
15	20:10:09	SELECT * FROM Customer_details LIMIT 0, 300	10 row(s) returned	0.000 sec / 0.000 sec
16	20:15:10	SELECT State, COUNT(*) AS Customercount FROM Customer GROUP BY State HAVING Customercount > 5	0 row(s) returned	0.000 sec / 0.000 sec
17	20:15:29	SELECT State, COUNT(*) AS Customercount FROM Customer GROUP BY State HAVING COUNT(*) > 5 LIMIT 0, 300	0 row(s) returned	0.000 sec / 0.000 sec
18	20:16:05	SELECT State, COUNT(*) AS Customercount FROM Customer GROUP BY State HAVING Customercount > 5	0 row(s) returned	0.000 sec / 0.000 sec

➤ QUERY THAT WILL RETURN THE NO OF CUSTOMERS IN EACH STATE, BASED ON THE "STATE" COLUMN IN THE "CUSTOMER_DETAILS" VIEW.



The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

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

SELECT State, COUNT(*) AS Customercount
FROM Customer_details
GROUP BY State;
```

The Results grid shows the output of the query:

State	Customercount
California	1
New York	1
Illinois	1
New South Wales	1
Berlin	1
Île-de-France	1
Tokyo	1
Western Cape	1
São Paulo	1
Telangana	1

➤ QUERY THAT RETURNS ALL THE COLUMNS FROM THE "CUSTOMER_DETAILS" VIEW, SORTED BY THE "STATE" COLUMN IN ASCENDING ORDER.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with various database objects. The main editor window contains a SQL query. The 'Result Grid' at the bottom shows the output of the query, which is a table with four columns: Full_name, Email, Phone_no, and State. The table contains ten rows of customer data.

```
102 SELECT State, COUNT(*) AS Customercount FROM Customer;
103 GROUP BY State;
104 HAVING Customercount > 5;
105
106 # 6. Write a query that will return the number of customers in each state, based on the "state" column in the "customer_details" view.
107
108 SELECT State, COUNT(*) AS Customercount
109 FROM Customer_details
110 GROUP BY State;
111
112 # 7. Write a query that returns all the columns from the "customer_details" view, sorted by the "state" column in ascending order.
113
114 SELECT * FROM Customer_details
115 ORDER BY State ASC;
116
```

Full_name	Email	Phone_no	State
Vikram Singh	vikram.singh@example.com	+49-15209876543	Berlin
John Doe	john.doe@example.com	+1-2029999999	California
Sanya Reddy	sanya.reddy@example.com	+33-612345678	Île-de-France
Michael Johnson	m.johnson@example.com	+1-202550188	Illinois
Asha Patel	asha.patel@example.com	+61-412345678	New South Wales
Jane Smith	jane.smith@example.com	+1-2026650188	New York
Manish Verma	manish.verma@example.com	+55-11987654321	São Paulo
Arjail Desai	arjail.desai@example.com	+91-9988776655	Telangana
Rohit Bose	rohit.bose@example.com	+81-9098765432	Tokyo
Sheha Gupta	sheha.gupta@example.com	+27-723456789	Western Cape