

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.

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'Schemas' section, there is a 'student' schema which contains a 'student' table. The SQL Editor pane displays the following SQL code:

```
-- Task:15/12/23
1 • CREATE DATABASE Student;
2 • USE Student;
3 • CREATE TABLE STUDENT (Roll_No INT PRIMARY KEY , Student_name Varchar(30), Marks INT, Grade CHAR(10));
4 • desc Student;
5 • INSERT INTO STUDENT VALUES
6     (1, 'Ankit', 88, 'B'),
7     (2, 'Kritik', 76, 'C'),
8     (3, 'Varna', 78, 'C'),
9     (4, 'Arun', 89, 'B'),
10    (5, 'Saranya', 95, 'A'),
11    (6, 'Lavanya', 68, 'D'),
12    (7, 'Viswa', 72, 'C'),
13    (8, 'Maya', 89, 'B'),
14    (9, 'Padma', 67, 'D'),
15    (10, 'Ravi', 92, 'A'));
16 • SELECT * FROM STUDENT;
17
18
19
```

The Output pane shows the results of the executed queries. It includes a table of log entries with columns for Time, Action, Message, Duration / Fetch, and a list of individual log entries:

Action	Output	Message	Duration / Fetch
USE Student;	4 rows(s) affected	0.000 sec	
CREATE TABLE STUDENT (Roll_No INT PRIMARY KEY , Student_name Varchar(30), Marks INT, Grade CHAR(10));	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec	
desc Student;	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.031 sec	
INSERT INTO STUDENT VALUES (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');	4 row(s) affected	0.016 sec	
SELECT * FROM STUDENT;	10 row(s) returned	0.000 sec / 0.000 sec	

(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;
18

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

```

STUDENT 3 X

Tables Views Stored Procedures Functions

student sys

Administration Schemas Information

Schema: student

Action Output

#	Time	Action	Message	Duration / Fetch
11	22:57:40	SELECT * FROM STUDENT LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
12	23:00:29	ALTER TABLE STUDENT ADD COLUMN(Contact Varchar(30))	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
13	23:03:54	ALTER TABLE STUDENT DROP COLUMN Grade	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
14	23:04:55	RENAME TABLE STUDENT TO CLASSTEN	0 row(s) affected	0.000 sec
15	23:06:12	TRUNCATE TABLE STUDENT	Error Code: 1146. Table 'student.student' doesn't exist	0.000 sec
16	23:06:29	TRUNCATE TABLE CLASSTEN	0 row(s) affected	0.031 sec
17	23:06:52	DROP TABLE CLASSTEN	0 row(s) affected	0.015 sec

Object Info Session

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

MySQL Workbench

Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL File 10* SQL File 3*

SCHEMAS

- constraints
- Tables
 - managers
 - Views
 - Defined Procedures
 - Functions
- sales
- Tables
 - orders
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- student
- sys
 - Tables
 - sys_config
 - Columns
 - Indexes
 - Foreign Keys

Administration Schemas

Information

Table: orders

Columns:

Order_id	int PK
Customer_name	varchar(30)
Product_name	varchar(50)
Ordered_Item	varchar(30)

Action Output

#	Time	Action	Message	Duration / Fetch
3	10:19:02	DROP DATABASE 'icschool'	1 row(s) affected	0.000 sec
4	10:19:12	DROP DATABASE 'joms'	3 row(s) affected	0.015 sec
5	10:20:00	CREATE TABLE Orders(Order_id INT PRIMARY KEY, Customer_name VARCHAR(30), Product_name VARCHAR(50) UNIQUE, Ordered_Item VARCHAR(30) UNIQUE NOT NULL);	0 row(s) affected	0.063 sec
6	10:20:00	USE Sales;	0 row(s) affected	0.000 sec
7	10:20:12	CREATE TABLE Orders(Order_id INT PRIMARY KEY, Customer_name VARCHAR(30), Product_name VARCHAR(50) UNIQUE, Ordered_Item VARCHAR(30) UNIQUE NOT NULL);	Error Code: 1050. Table 'orders' already exists	0.000 sec
8	10:25:43	DESC Sales	Error Code: 1146. Table 'sales.sales' doesn't exist	0.000 sec
9	10:26:05	DESC Orders	4 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

10:26 AM 12/22/2023

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.

6 • CREATE TABLE Orders(Order_id INT PRIMARY KEY, Customer_name VARCHAR(30) UNIQUE NOT NULL, Product_name VARCHAR(50), Ordered_Item VARCHAR(30) NOT NULL);

7 • ALTER TABLE Orders ADD COLUMN(Order_quantity int NOT NULL);

8 • DESC ORDERS;

9

10 • RENAME TABLE Orders to Sales_orders;

11 • DESC Sales_orders;

12 • INSERT INTO Sales_orders VALUES

(001, 'Aditya', 'Samsung', 'Smart_phone', 1),
(002, 'Mariya', 'Apple', 'Laptop', 2),
(003, 'Akshita', 'HP', 'Laptop', 1),
(004, 'Kartik', 'Dell', 'Laptop', 5),
(005, 'Anzar', 'Apple', 'Iphone', 1),
(006, 'Mathew', 'Lenovo', 'Tablet', 3),
(007, 'Sridevi', 'HP', 'Printer', 10),
(008, 'Nidhi', 'Samsung', 'Speaker', 1),
(009, 'Arun', 'JBL', 'Soundbar', 1),
(010, 'Manu', 'Sony', 'Camera', 2);

Output

Action Output

#	Time	Action	Message	Duration / Fetch
29	11:56:18	DESC Sales_orders	5 row(s) returned	0.000 sec / 0.000 sec
30	12:14:57	INSERT INTO Sales_orders VALUES(001, 'Aditya', 'Samsung', 'Smart_phone'), (002, 'Mariya', 'Apple', 'Laptop'), ...	Error Code: 1136. Column count doesn't match value count at row 1	0.015 sec
31	12:15:54	DESC Sales_orders	5 row(s) returned	0.000 sec / 0.000 sec
32	12:16:24	INSERT INTO Sales_orders VALUES(001, 'Aditya', 'Samsung', 'Smart_phone'), (002, 'Mariya', 'Apple', 'Laptop'), ...	Error Code: 1136. Column count doesn't match value count at row 1	0.000 sec
33	12:17:34	EXPLAIN INSERT INTO Sales_orders VALUES (001, 'Aditya', 'Samsung', 'Smart_phone'), (002, 'Mariya', 'Apple', 'Laptop')	OK	0.000 sec
34	12:17:34	EXPLAIN FORMAT=JSON INSERT INTO Sales_orders VALUES (001, 'Aditya', 'Samsung', 'Smart_phone'), (002, 'Mariya', 'Apple', 'Laptop')	... OK	0.000 sec
35	12:18:10	INSERT INTO Sales_orders VALUES (001, 'Aditya', 'Samsung', 'Smart_phone'), (002, 'Mariya', 'Apple', 'Laptop')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec

4. Retrieve customer_name and Product_name from the sales_orders table.

The screenshot shows a database interface with the following details:

- SQL History:** Contains two queries:
 - 25 • SELECT * FROM Sales_orders;
 - 24 • SELECT Customer_name, Product_name FROM Sales_orders;
- Result Grid:** Displays the results of the second query:

Customer_name	Product_name
Maruya	Apple
Akshita	HP
Kartik	Dell
Anzar	Apple
Mathew	Lenovo
Sridevi	HP
Nidhi	Samsung
Arun	JBL
Manu	Sony
- Action Output:** Shows the execution history of the last 10 actions, including insertions, a select, and an update command.

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

The screenshot shows a database interface with the following details:

- Tables:** Shows the structure of the sales_orders table with columns: Order_id, Customer_name, Product_name, Ordered_Item, and Order_quantity.
- SQL History:** Contains three queries:
 - 26 • UPDATE Sales_orders SET Product_name='Apple' WHERE Order_id=004;
 - 27 • SELECT * FROM Sales_orders;
- Result Grid:** Displays the results of the second query after the update:

Order_id	Customer_name	Product_name	Ordered_Item	Order_quantity
1	Aditya	Samsung	Smart_phone	1
2	Maruya	Apple	Laptop	2
3	Akshita	HP	Laptop	1
4	Kartik	Apple	Laptop	5
5	Anzar	Apple	Iphone	1
6	Mathew	Lenovo	Tablet	3
7	Sridevi	HP	Printer	10
8	Nidhi	Samsung	Speaker	1
9	Arun	JBL	Soundbar	1
10	Manu	Sony	Camera	2
- Action Output:** Shows the execution history of the last 10 actions, including the update command.

6. Delete the sales_orders table from the database.

```

27 • SELECT * FROM Sales_orders;
28
29 -- Delete table from the database
30 • DROP TABLE Sales_orders;
31
32
33
34

```

Output:

Action	Time	Action	Message	Duration / Fetch
37	12-23-01	SELECT * FROM Sales_orders LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 st
38	12-23-16	SELECT Customer_name and Product_name from Sales_orders LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 st
39	12-24-43	SELECT Customer_name, Product_name from Sales_orders LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 st
40	12-28-51	UPDATE Sales_orders SET Product_name='Apple' WHERE Order_id=004	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
41	12-29-06	SELECT * FROM Sales_orders LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 st
42	12-31-24	DROP TABLE Sales_order	Error Code: 1051. Unknown table 'sales.sales_order'	0.000 sec
43	12-31-48	DROP TABLE Sales_orders	0 row(s) affected	0.031 sec

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),
9 (103, 'Canada', 350000, 2100)

```

```

18  # 'Create a table Persons
19
20 • CREATE TABLE PERSONS(Id INT PRIMARY KEY, Fname VARCHAR(30), Lname VARCHAR(30), Population INT, Rating INT, Country_Id INT, Country_name VARCHAR(30));
21 • INSERT INTO PERSONS VALUES
22   (201, 'John', 'Doe', 50000, 4, 101, 'Australia'),
23   (202, 'Jane', 'Smith', 70000, 5, 102, 'Belize'),
24   (203, 'Bob', 'Johnson', 80000, 4, 103, 'Canada'),
25   (204, 'Alice', 'Williams', 60000, 3, 104, 'Denmark'),
26   (205, 'Charlie', 'Brown', 45000, 5, 105, 'England')

```

Inner join

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Shows the database structure with tables `country` and `persons`.
- Table: country** (selected):

ID	COUNTRY_name	Population	Area
201	Australia	50000	4
202	Belize	70000	5
203	Canada	80000	4
204	Denmark	60000	3
205	England	45000	5
206	Ireland	55000	4
207	Jordan	30000	3
208	Kenya	40000	5
209	Martin	75000	4
210	Taylor	90000	5
- Table: persons** (selected):

ID	Fname	Lname	Population	Rating
201	John	Doe	50000	4
202	Jane	Smith	70000	5
203	Bob	Johnson	80000	4
204	Alice	Williams	60000	3
205	Charlie	Brown	45000	5
206	Eva	Anderson	55000	4
207	George	Miller	30000	3
208	Ivy	Martin	40000	5
209	Jack	Clark	75000	4
210	Karen	Taylor	90000	5
- Result Grid:** Shows the joined data from both tables.
- Action Output:** Displays the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	11:19:24	USE Jobs	0 row(s) affected	0.000 sec / 0.000 sec
2	11:21:01	SELECT * FROM Persons INNER JOIN Country ON Persons.Country_id=Country.Id LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
3	11:22:31	SELECT Id, Fname, Lname, Population, Rating FROM Persons INNER JOIN Country ON Persons.Country_id=C...	Error Code: 1052. Column 'Id' in field list is ambiguous	0.016 sec
4	11:22:53	SELECT Persons.Id, Fname, Lname, Population, Rating FROM Persons INNER JOIN Country ON Persons.Cou...	Error Code: 1052. Column 'Population' in field list is ambiguous	0.000 sec
5	11:23:34	SELECT Persons.Id, Fname, Lname, Persons.Population, Rating FROM Persons INNER JOIN Country ON Perso...	10 row(s) returned	0.000 sec / 0.000 sec

Left join

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Shows the database structure with tables `country` and `persons`.
- Table: persons** (selected):

ID	Fname	Lname	Population	Rating	Country_Id	Country_name
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
205	Charlie	Brown	45000	5	105	England
- Table: country** (selected):

ID	COUNTRY_name	Population	Area
201	Australia	50000	4
202	Belize	70000	5
203	Canada	80000	4
204	Denmark	60000	3
205	England	45000	5
206	Ireland	55000	4
207	Jordan	30000	3
208	Kenya	40000	5
209	Martin	75000	4
210	Taylor	90000	5
- Result Grid:** Shows the joined data from both tables.
- Action Output:** Displays the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
10	12:59:00	SELECT Persons.Id, Fname, Lname, Persons.Population FROM PERSONS LEFT JOIN Country ON Persons...	10 row(s) returned	0.000 sec / 0.000 sec
11	13:03:07	SELECT DISTINCT Country_id FROM Country WHERE Country_id IS NOT NULL AND Country_id NOT IN (SE...	0 row(s) returned	0.000 sec / 0.000 sec
12	13:03:49	SELECT DISTINCT Persons.Country_id FROM PERSONS WHERE Persons.Country_id IS NOT NULL AND ...	0 row(s) returned	0.000 sec / 0.000 sec
13	13:16:41	SELECT Persons.Id, Fname, Lname, Persons.Population FROM PERSONS LEFT JOIN Country ON Persons...	10 row(s) returned	0.015 sec / 0.000 sec
14	13:17:44	SELECT * FROM PERSONS LEFT JOIN COUNTRY ON PERSONS.Country_id = COUNTRY.Id LIMIT 0, 1000	Error Code: 1146. Table 'joins.personsleft' doesn't exist	0.000 sec
15	13:17:52	SELECT * FROM PERSONS LEFT JOIN COUNTRY ON PERSONS.Country_id = COUNTRY.Id LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
16	13:20:42	SELECT Persons.Id AS Person_Id, Persons.Fname, Persons.Lname, Persons.Population AS Person...	10 row(s) returned	0.015 sec / 0.000 sec

Right join

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the 'persons' schema with its tables, columns, indexes, and triggers. A 'Table: country' node is selected. Below it, the 'Information' node is expanded, showing the 'Columns' section for the 'country' table with four columns: 'Id' (int PK), 'COUNTRY_name' (varchar(30)), 'Population' (int), and 'AREA' (int).

The main pane contains a T-SQL query:

```

48
49 -- Right join
50 • SELECT Persons.Id AS Person_Id,
51     Persons.Fname,
52     Persons.Lname,
53     Persons.Population AS Person_Population,
54     Persons.Country_Id,
55     Country.Country_name,
56     Country.Population AS Country_Population,
57     Country.AREA
58 FROM Country RIGHT JOIN
59     Persons ON Country.Id=Persons.Country_Id;
60

```

The 'Output' tab shows the execution log with several rows of activity, including SELECT statements and an error message about a non-existent table 'person:left'. The status bar at the bottom right indicates the time is 1:30 PM and the date is 1/19/2024.

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.

The screenshot shows the SQL Server Management Studio interface. The code in the query window is:

```

63 -- Write an SQL query to print the first three characters of Country_name from the Country table.
64 • SELECT
65     Country_name,
66     SUBSTRING(Country_name, 1, 3) AS First_Three_Characters
67 FROM Countries;
68
69
70

```

The results grid displays the following data:

Country_name	First_Three_Characters
Australia	Aus
Belize	Bel
Canada	Can
Denmark	Den
Estonia	Est
Finland	Fin
Ghana	Gha
Ireland	Ire
Jordan	Jor
Kenya	Ken

The 'Output' tab shows the execution log with three rows of activity, including the use of joins and the execution of the SELECT statement.

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

FROM Country;

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

SELECT Fname, Lname, CONCAT(Fname,Lname) AS Full_name FROM Persons;

Fname	Lname	Full_name
John	Doe	JohnDoe
Jane	Smith	JaneSmith
Bob	Johnson	BobJohnson
Alice	Williams	AliceWilliams
Charlie	Brown	CharlieBrown
Eva	Anderson	EvaAnderson
George	Miller	GeorgeMiller
Ivy	Martin	IvyMartin
Jack	Clark	JackClark
Karen	Taylor	KarenTaylor

Action Output

#	Time	Action	Message	Duration / Fetch
1	11:40:49	USE Joms	0 row(s) affected	0.000 sec
2	11:50:08	SELECT Country_name, LEFT(Country_name, 3) AS First_Three_Characters FROM Country LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 sec
3	11:51:01	SELECT Country_name, SUBSTRING(Country_name, 1, 3) AS First_Three_Characters FROM Country LIM...	10 row(s) returned	0.000 sec / 0.000 sec
4	11:57:34	SELECT PERSONS, CONCAT(Fname,Lname) AS Full_name LIMIT 0, 1000	Error Code: 1054. Unknown column 'PERSONS' in field list'	0.015 sec
5	11:58:12	SELECT Fname, Lname, CONCAT(Fname,Lname) AS Full_name FROM Persons LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

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

SELECT DISTINCT Count(Country_name) FROM Persons as Unique_values;

Count(Country_name)
10

Action Output

#	Time	Action	Message	Duration / Fetch
2	11:50:08	SELECT Country_name, LEFT(Country_name, 3) AS First_Three_Characters FROM Country LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 sec
3	11:51:01	SELECT Country_name, SUBSTRING(Country_name, 1, 3) AS First_Three_Characters FROM Country LIM...	10 row(s) returned	0.000 sec / 0.000 sec
4	11:57:34	SELECT PERSONS, CONCAT(Fname,Lname) AS Full_name LIMIT 0, 1000	Error Code: 1054. Unknown column 'PERSONS' in field list'	0.015 sec
5	11:58:12	SELECT Fname, Lname, CONCAT(Fname,Lname) AS Full_name FROM Persons LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
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

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.

```

75 -- Write a query to print the maximum population from the Country table.
76 • SELECT MAX(Population) AS Max_Population FROM Country;
77 -- Write a query to print the minimum population from the Country table.
78 • SELECT MIN(Population) AS Min_Population FROM Countries;
```


```

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

```

/*
80 -- Insert 2 new rows to the Persons table making the Lname NULL. Then write another query to count Lname from Persons table.
81 • INSERT INTO Persons VALUES
82 (215, 'Miriam', Null, 45000, 5, 105, 'Estonia'),
83 (216, 'Nate', Null, 40000, 5, 108, 'Ireland');
84 • SELECT COUNT(Lname) AS Count_Lname FROM Persons;
```


```

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.

```

84 •   SELECT COUNT(Lname) AS Count_Lname FROM Persons;
85
86 -- Write a query to find the number of rows in the Persons table.
87 •   SELECT COUNT(*) AS number_of_rows FROM PERSONS;
88
89 -- Write an SQL query to return the current date and time.
90 •   SELECT NOW();
91

```

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

NOW() □ 2024-01-24 12:58:34

Result 2 ×

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:51:50	USE Joins	0 row(s) affected	0.000 sec
2	12:57:16	SELECT COUNT(*) AS number_of_rows FROM PERSONS LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	12:58:34	SELECT NOW() LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec

Object Info Session 12:58 PM

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)

```

98 •   SELECT NOW();
99
100 -- Write an SQL query to show the population of the Country table for the first 3 rows. (Hint: Use LIMIT)
101 •   SELECT Country_name,Population FROM COUNTRY LIMIT 3;
102
103 -- Write a query to print 3 random rows of countries. (Hint: Use rand() function and LIMIT)
104 •   SELECT * FROM Country ORDER BY RAND() LIMIT 3;
105

```

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

Id	COUNTRY_name	Population	AREA
101	Australia	300000	1200
109	Jordan	230000	789
104	Denmark	230000	1100

Country 6 ×

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:51:50	USE Joins	0 row(s) affected	0.000 sec
2	12:57:16	SELECT COUNT(*) AS number_of_rows FROM PERSONS LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	12:58:34	SELECT NOW() LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
4	13:03:30	SELECT Population FROM COUNTRY LIMIT 3	3 row(s) returned	0.000 sec / 0.000 sec
5	13:03:55	SELECT Country_name,Population FROM COUNTRY LIMIT 3	3 row(s) returned	0.000 sec / 0.000 sec
6	13:04:48	SELECT RAND() FROM Country LIMIT 3	3 row(s) returned	0.000 sec / 0.000 sec
7	13:06:49	SELECT * FROM Country ORDER BY RAND() LIMIT 3	3 row(s) returned	0.000 sec / 0.000 sec

Object Info Session 1:06 PM

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.

```

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

12
114      -- Write a user-defined function to calculate age using DOB.
114 DELIMITER $$

115 • CREATE FUNCTION AGE(DOB Date)
116     RETURNS int
117     DETERMINISTIC
118     BEGIN
119         DECLARE age INT;
120         SET age=TIMESTAMPDIFF(Year, DOB, NOW());
121         RETURN Age;
122     END $$

123     DELIMITER ;
124 • -- Write a select query to fetch the Age of all persons reusing the function that has been created.
125     SELECT Id, Fname, Lname, Country_name, DOB, Age(DOB) AS Current_Age FROM Persons;
126

```

Result Grid						
	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
])	210	Karen	Taylor	Kenya	1992-05-03	31

Task 7:[Subqueries]

Questions:

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

- Find the number of persons in each country.

```

126      -- Find the number of persons in each country.
127 •  SELECT Country.Country_name, (SELECT Count(*) FROM Persons WHERE Persons.Country_id=Country.Id) FROM Country;
128

```

Result Grid

Country_name	(SELECT Count(*) FROM Persons WHERE Persons.Country_id=Country.Id)
Australia	1
Belize	1
Canada	1
Denmark	1
Estonia	2
Finland	1
Ghana	1
Ireland	2
Jordan	1

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:14:20	USE Joins	0 row(s) affected	0.000 sec
2	12:16:17	SELECT SUM(Population) AS Total_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.
3	12:16:56	SELECT * FROM COUNTRY LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.
4	12:35:20	SELECT Country.Country_name, (SELECT Count(*) FROM Persons WHERE Persons.Country_id=Country.Id) FR...	10 row(s) returned	0.000 sec / 0.

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

```

try
Columns
Indexes
Foreign Keys
Triggers
rions
Procedures
ts
schemas

```

Result Grid

Country_name	Persons_count
Estonia	2
Ireland	2
Australia	1
Belize	1
Canada	1
Denmark	1
Finland	1
Ghana	1
Jordan	1
Venice	1

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:14:20	USE Joins	0 row(s) affected	0.000 sec
2	12:16:17	SELECT SUM(Population) AS Total_Population FROM Country LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
3	12:16:56	SELECT * FROM COUNTRY LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
4	12:35:20	SELECT Country.Country_name, (SELECT Count(*) FROM Persons WHERE Persons.Country_id=Country.Id) AS Persons_count FROM Country ORDER BY Persons_count DESC;	10 row(s) returned	0.000 sec / 0.000 sec
5	12:40:21	SELECT Country.Country_name, (SELECT Count(*) FROM Persons WHERE Persons.Country_id=Country.Id) AS...	Error Code: 3029. Expression #1 of ORDER BY contains aggregate function and applies to the result of a non-aggregat...	0.000 sec
6	12:42:03	SELECT Country.Country_name, (SELECT Count(*) FROM Persons WHERE Persons.Country_id=Country.Id) AS...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server...	0.016 sec
7	12:42:47	SELECT Country.Country_name, (SELECT Count(*) FROM Persons WHERE Persons.Country_id=Country.Id) AS...	10 row(s) returned	0.016 sec / 0.000 sec

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

Tables
 ▶ country
 ▶ Columns
 ▶ Indexes
 ▶ Foreign Keys
 ▶ Triggers
 persons

Views
 Stored Procedures
 Functions
 Administration Schemas
 nation

Table: persons

Columns:

ID	Fname	Lname	Population	Rating	Country_Id	Country_name	DOB
int PK	varchar(30)	varchar(30)	int	int	int	varchar(30)	date

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

```

132
133 -- Find out an average rating for Persons in respective countries if the average is greater than 3.0
134 • SELECT Fname, Lname, Country_name, AVG(Rating) AS Avg_rating FROM Persons WHERE
135     Country_name IN (SELECT Country_name FROM Persons GROUP BY Country_name HAVING AVG(Rating) > 3.0) GROUP BY Fname, Lname, Country_name;
+---+
| Fname | Lname | Country_name | Avg_rating |
| John  | Doe   | Australia    | 4.0000    |
| Jane  | Smith | Belize       | 5.0000    |
| Bob   | Johnson | Canada      | 4.0000    |
| Charlie | Brown | Estonia      | 5.0000    |
| Eva   | Anderson | Finland     | 4.0000    |
| Ivy   | Martin | Ireland      | 5.0000    |
| Jack  | Clark | Jordan       | 4.0000    |
| Karen | Taylor | Kenya        | 5.0000    |
| Miriam | null | Estonia      | 5.0000    |
| Niall | null | Ireland      | 5.0000    |
+---+

```

Result 2 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	11:24:08	USE Joins	0 row(s) affected	0.000 sec
2	12:33:04	SELECT Fname, Lname, Country_name, Avg(Rating) FROM Persons WHERE(SELECT Rating> 3 FROM Person...	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregat...	0.000 sec
3	12:42:06	SELECT Fname, Lname, Country_name, Avg(Rating) AS Avg_rating FROM Persons WHERE(SELECT Rating> 3...	Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column...	0.000 sec
4	12:51:07	SELECT Fname, Lname, Country_name, AVG(Rating) AS Avg_rating FROM Persons WHERE Co...	10 row(s) returned	0.000 sec / 0.000
5	12:52:21	SELECT Fname, Lname, Country_name, AVG(Rating) AS Avg_rating FROM Persons WHERE Co...	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregat...	0.000 sec
6	12:54:18	SELECT Fname, Lname, Country_name, Avg(Rating) AS Avg_rating FROM Persons WHERE Co...	Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column...	0.000 sec
7	12:54:41	SELECT Fname, Lname, Country_name, Avg(Rating) AS Avg_rating FROM Persons WHERE Co...	10 row(s) returned	0.000 sec / 0.000
7	12:54:41	SELECT Fname, Lname, Country_name, Avg(Rating) AS Avg_rating FROM Persons WHERE Co...	10 row(s) returned	0.000 sec / 0.000

Info Session

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

Tables
 ▶ Columns
 ▶ Indexes
 ▶ Foreign Keys
 ▶ Triggers
 persons

Views
 Stored Procedures
 Functions
 Administration Schemas
 Information

Table: persons

Columns:

ID	Fname	Lname	Population	Rating	Country_Id	Country_name	DOB
int PK	varchar(30)	varchar(30)	int	int	int	varchar(30)	date

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

```

137 -- Find the countries with the same rating as the USA. (Use Subqueries). Checking for Ireland here as USA is not there.
138 • SELECT DISTINCT Country_id, Country_name, Rating FROM Persons WHERE Rating IN (SELECT Rating FROM Persons WHERE Country_name='Ireland');
+---+
| Country_id | Country_name | Rating |
| 102        | Belize       | 5       |
| 105        | Estonia      | 5       |
| 108        | Ireland      | 5       |
| 110        | Kenya        | 5       |
+---+

```

Persons 3 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
4	12:51:07	SELECT Fname, Lname, Country_name, AVG(Rating) AS Avg_rating FROM Persons WHERE ... 10 row(s) returned	0.000 sec / 0.000 sec	
5	12:52:21	SELECT Fname, Lname, Country_name, AVG(Rating) AS Avg_rating FROM Persons WHERE ... Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregat...	0.000 sec	
6	12:54:18	SELECT Fname, Lname, Country_name, Avg(Rating) AS Avg_rating FROM Persons WHERE Country_name... Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated col...	0.000 sec	
7	12:54:41	SELECT Fname, Lname, Country_name, Avg(Rating) AS Avg_rating FROM Persons WHERE Country_name... 10 row(s) returned	0.000 sec / 0.000 sec	
8	13:35:40	SELECT DISTINCT Country_id, Country_name, Rating FROM Persons WHERE (SELECT Rating FROM Perso... Error Code: 1242. Subquery returns more than 1 row	0.000 sec	
9	13:37:12	SELECT DISTINCT Country_id, Country_name, Rating FROM Persons WHERE Rating=(SELECT Rating F... Error Code: 1242. Subquery returns more than 1 row	0.000 sec	
10	13:38:13	SELECT DISTINCT Country_id, Country_name, Rating FROM Persons WHERE Rating IN (SELECT Rating F... 4 row(s) returned	0.016 sec / 0.000 sec	

Object Info Session

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

```

139
140      -- Select all countries whose population is greater than the average population of all nations.
141 •   SELECT Country_Id, Country_name, Population FROM Persons WHERE Population > (SELECT Avg(Population) FROM Persons) GROUP BY Country_Id, Country_name, Population;
142

Result Grid | Filter Rows: Export: Wrap Cell Content: 15
Result Grid
Form Editor
Field Types
Read Only

30)
30) Persons 5 x
30) Output: Action Output
# Time Action
8 13:35:40 SELECT DISTINCT Country_id, Country_name, Rating FROM Persons WHERE (SELECT Rating FROM Perso... Error Code: 1242. Subquery returns more than 1 row 0.000 sec
9 13:37:12 SELECT DISTINCT Country_id, Country_name, Rating FROM Persons WHERE Rating=(SELECT Rating FRO... Error Code: 1242. Subquery returns more than 1 row 0.000 sec
10 13:38:13 SELECT DISTINCT Country_id, Country_name, Rating FROM Persons WHERE Rating IN (SELECT Rating F... 4 row(s) returned 0.016 sec / 0.000 sec
11 13:45:46 SELECT Country_id, Country_name, Population, Avg(Population) as Avg_population FROM Persons WHERE ... Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggrega... 0.000 sec
12 13:46:06 SELECT Country_id, Country_name, Population, Avg(Population) as Avg_population FROM Persons WHERE ... Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated colu... 0.000 sec
13 13:46:38 SELECT Country_id, Country_name, Population, Avg(Population) as Avg_population FROM Persons WHERE ... 5 row(s) returned 0.000 sec / 0.000 sec
14 13:47:07 SELECT Country_id, Country_name, Population FROM Persons WHERE Population > (SELECT Avg(Popula... 5 row(s) returned 0.000 sec / 0.000 sec

147 PM 1/31/2024

```

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

The screenshot shows the MySQL Workbench interface with the following details:

- File** → Local instance MySQL80
- Navigator**: Schemas (selected), bank, constraints, func, joins, Tables (selected), country, persons, Views, Stored Procedures, Functions, orgdb, sales, student, sys.
- SCHEMAS**: Filter objects (empty).
- SQL Editor**: SQL File 10* (selected), SQL File 3*, SQL Clauses*, Bank data*, SQL Functions*, SQL VIEW*, Joins*, Views*. Limit to 1000 rows. The code is a database creation script for a 'PRODUCT' database, creating tables for Customer and Product.
- Information**: Administration, Schemas.
- No object selected**.
- Action Output**: Shows the execution history with the following entries:

#	Time	Action	Message	Duration / Fetch
5	12/06/25	INSERT INTO customer VALUES (1, 'John', 'Doe', 'john.doe@example.com', 1234567890, '123 Main St', 'Anytown')	Error Code: 1264. Out of range value for column 'Phone_no' at row 3	0.000 sec
6	12/09/06	INSERT INTO customer VALUES (1, 'John', 'Doe', 'john.doe@example.com', 1234567890, '123 Main St', 'Anytown')	Error Code: 1264. Out of range value for column 'Phone_no' at row 3	0.016 sec
7	12/09/18	DROP Table Customer	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server ...	0.000 sec
8	12/09/56	DROP Table Customer	0 row(s) affected	0.000 sec
9	12/10/24	CREATE TABLE Customer (Customer_Id INT PRIMARY KEY, First_name VARCHAR(30), Last_name VARCHAR(30), Email VARCHAR(30), Phone_no INT, Address VARCHAR(100), City VARCHAR(30), State VARCHAR(30), Zipcode INT, Country VARCHAR(30))	0 row(s) affected	0.031 sec
10	12/10/34	INSERT INTO customer VALUES (1, 'John', 'Doe', 'john.doe@example.com', 1234567890, '123 Main St', 'Anytown')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.015 sec

- Object Info** and **Session** tabs are visible at the bottom.

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.

The screenshot shows the MySQL Workbench interface with the following details:

- File**, **Edit**, **View**, **Query**, **Database**, **Server**, **Tools**, **Scripting**, **Help** menu.
- Schemas** pane on the left, currently expanded to show **JOINS** under **Tables**.
- SQL Editor** pane with the following SQL code:

```
16 (7, 'Grace', 'Taylor', 'grace.taylor@example.com', '7778899999', '404 Cedar St', 'Somewhere', 'NY', 45678, 'Germany'),
17 (8, 'David', 'Anderson', 'david.anderson@example.com', '6667778888', '505 Pine St', 'Anywhere', 'FL', 56789, 'US'),
18 (9, 'Olivia', 'Brown', 'olivia.brown@example.com', '2233344444', '606 Oak St', 'Everywhere', 'CA', 78901, 'US'),
19 (10, 'Sam', 'Clark', 'sam.clark@example.com', '3344455555', '707 Maple St', 'Nowhere', 'TX', 12345, 'US');

20 -- 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.
21 CREATE VIEW Customer_info AS SELECT CONCAT(first_name, last_name) AS Full_name, Email FROM Customer;
22 *
23 * SELECT * FROM Customer_info;
24
```
- Result Grid** pane showing the results of the query:

Full_name	Email
JohnDoe	john.doe@example.com
JaneSmith	jane.smith@example.com
AliceJohnson	alice.johnson@example.com
BobWilliams	bob.williams@example.com
EvaMiller	eva.miller@example.com
CharlieDavis	charlie.davis@example.com
GraceTaylor	grace.taylor@example.com
DavidAnderson	david.anderson@example.com
OliviaBrown	olivia.brown@example.com
SamClark	sam.clark@example.com

- Output** pane showing the history of actions:

#	Time	Action	Message	Duration / Fetch
6	12:09:06	INSERT INTO customer VALUES (1, 'John', 'Doe', 'john.doe@example.com', 1234567890, '123 Main St', 'Any...', Error Code: 1264. Out of range value for column 'Phone_no' at row 3)	Error Code: 1264. Out of range value for column 'Phone_no' at row 3	0.016 sec
7	12:09:18	DROP Customer	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Customer' at line 1	0.000 sec
8	12:09:56	DROP Table Customer	0 row(s) affected	0.000 sec
9	12:10:24	CREATE TABLE Customer (Customer_Id INT PRIMARY KEY, first_name VARCHAR(30), last_name VARCHAR(30), phone_no INT, address VARCHAR(100), city VARCHAR(50), state CHAR(2), zip_code INT, email VARCHAR(100))	0 row(s) affected	0.031 sec
10	12:10:34	INSERT INTO customer VALUES (1, 'John', 'Doe', 'john.doe@example.com', 1234567890, '123 Main St', 'Any...', Error Code: 1264. Out of range value for column 'Phone_no' at row 3)	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.015 sec
11	12:15:37	CREATE VIEW Customer_info AS SELECT CONCAT(first_name, last_name) AS Full_name, Email FROM Customer	0 row(s) affected	0.000 sec
12	12:16:00	SELECT * FROM Customer_info LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

- Object Info** and **Session** tabs at the bottom.

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

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** Shows tables like bank, country, persons, Views, and Stored Procedures.
- Code Editor:**

```

25 -- Create a view named US_Customers that displays customers located in the US.
26 • CREATE VIEW US_Customers AS SELECT * FROM CUSTOMER WHERE COUNTRY = 'US';
27 • SELECT * FROM US_Customers;
28

```
- Result Grid:** Displays the results of the SELECT query from the US_Customers view. The columns are Customer_Id, First_name, Last_name, Email, Phone_no, Address, City, State, Zipcode, and Country. The data includes 10 rows of customer information.
- Action Output:** Shows the history of actions taken in the session, including the creation of the view and its selection.
- System Bar:** Shows the date and time as 12:20 PM 2/1/2024.

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

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** Shows tables like bank, country, persons, Views, and Stored Procedures.
- Code Editor:**

```

25 -- Create a view named US_Customers that displays customers located in the US.
26 • CREATE VIEW US_Customers AS SELECT * FROM CUSTOMER WHERE COUNTRY = 'US';
27 • SELECT * FROM US_Customers;
28
29 -- Create another view named Customer_details with columns full name(Combine first_name and last_name), email, phone_no, and state.
30 • CREATE VIEW Customer_details AS SELECT CONCAT(first_name, last_name) AS Full_name, email, Phone_no, State FROM Customer;
31 • SELECT * FROM Customer_details;
32

```
- Result Grid:** Displays the results of the SELECT query from the Customer_details view. The columns are Full_name, email, Phone_no, and State. The data includes 10 rows of customer information.
- Action Output:** Shows the history of actions taken in the session, including the creation of the view and its selection.
- System Bar:** Shows the date and time as 12:24 PM 2/1/2024.

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

MySQL Workbench Local instance MySQL80

```

SELECT * FROM Customer_details;
-- Update phone numbers of customers who live in California for Customer_details view.
-- Disable safe update mode
SET SQL_SAFE_UPDATES = 0;
-- Update phone numbers of customers who live in California for Customer_details view
UPDATE Customer_details SET Phone_no = '1456789' WHERE STATE='CA';

```

Full_name	email	Phone_no	State
JohnDoe	john.doe@example.com	1456789	CA
JaneSmith	jane.smith@example.com	165789010	NY
AliceJohnson	alice.johnson@example.com	5551234567	TX
BobWilliams	bob.williams@example.com	112233333	FL
EvaMiller	eva.miller@example.com	1456789	CA
CharlieDavis	charlie.davis@example.com	444555666	TX
GraceTaylor	grace.taylor@example.com	777888999	NY
DavidAnderson	david.anderson@example.com	666777888	FL
OliviaBrown	olivia.brown@example.com	1456789	CA
Customer_details	customer_details	111111111	TX

Action Output

#	Time	Action	Message	Duration / Fetch
22	12:34:54	SELECT * FROM Customer_details LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
23	13:02:02	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.016 sec
24	13:02:11	UPDATE Customer_details SET Phone_no = '1456789' WHERE STATE='CA'	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a column in an UPDATE clause.	0.000 sec
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

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.

MySQL Workbench Local instance MySQL80

```

-- Count the number of customers in each state and show only states with more than 5 customers
SELECT State, COUNT(*) AS Customer_count FROM Customer_details GROUP BY State HAVING Customer_count>5;

-- 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 Customer_count FROM Customer_details GROUP BY State;

```

State	Customer_count
CA	3
NY	2
TX	3
FL	2

Action Output

#	Time	Action	Message	Duration / Fetch
24	13:02:11	UPDATE Customer_details SET Phone_no = '1456789' WHERE STATE='CA'	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a column in an UPDATE clause.	0.000 sec
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_count>5;	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

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

```

The screenshot shows the execution of the provided SQL code. The results are displayed in a grid:

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	2223334444	TX

Customer_details 9

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Read Only

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');
74 ●  CALL InsertWorker(4, 'Mira', 6500, 'Analytics');
75 ●  CALL InsertWorker(5, 'Raj', 7800, 'Finance');
76 ●  CALL InsertWorker(6, 'Lata', 7200, 'Finance');
77 ●  CALL InsertWorker(7, 'Shivani', 7500, 'Finance');
78 ●  CALL InsertWorker(8, 'Vikas', 7800, 'Finance');
79 ●  CALL InsertWorker(9, 'Nisha', 7200, 'Finance');
80 ●  CALL InsertWorker(10, 'Rajesh', 7500, 'Finance');
81 ●  CALL InsertWorker(11, 'Kiran', 7800, 'Finance');
82 ●  CALL InsertWorker(12, 'Rakesh', 7200, 'Finance');
83 ●  CALL InsertWorker(13, 'Suresh', 7500, 'Finance');
84 ●  CALL InsertWorker(14, 'Rajesh', 7800, 'Finance');
85 ●  CALL InsertWorker(15, 'Kiran', 7200, 'Finance');
86 ●  CALL InsertWorker(16, 'Rakesh', 7500, 'Finance');
87 ●  CALL InsertWorker(17, 'Suresh', 7800, 'Finance');
88 ●  CALL InsertWorker(18, 'Rajesh', 7200, 'Finance');
89 ●  CALL InsertWorker(19, 'Kiran', 7500, 'Finance');
90 ●  CALL InsertWorker(20, 'Rakesh', 7800, 'Finance');
91 ●  CALL InsertWorker(21, 'Suresh', 7200, 'Finance');
92 ●  CALL InsertWorker(22, 'Rajesh', 7500, 'Finance');
93 ●  CALL InsertWorker(23, 'Kiran', 7800, 'Finance');
94 ●  CALL InsertWorker(24, 'Rakesh', 7200, 'Finance');
95 ●  CALL InsertWorker(25, 'Suresh', 7500, 'Finance');
96 ●  CALL InsertWorker(26, 'Rajesh', 7800, 'Finance');
97 ●  CALL InsertWorker(27, 'Kiran', 7200, 'Finance');
98 ●  CALL InsertWorker(28, 'Rakesh', 7500, 'Finance');
99 ●  CALL InsertWorker(30, 'Suresh', 7800, 'Finance');
100 ●  CALL InsertWorker(31, 'Rajesh', 7200, 'Finance');
101 ●  CALL InsertWorker(32, 'Kiran', 7500, 'Finance');
102 ●  CALL InsertWorker(33, 'Rakesh', 7800, 'Finance');
103 ●  CALL InsertWorker(34, 'Suresh', 7200, 'Finance');
104 ●  CALL InsertWorker(35, 'Rajesh', 7500, 'Finance');
105 ●  CALL InsertWorker(36, 'Kiran', 7800, 'Finance');
106 ●  CALL InsertWorker(37, 'Rakesh', 7200, 'Finance');
107 ●  CALL InsertWorker(38, 'Suresh', 7500, 'Finance');
108 ●  CALL InsertWorker(39, 'Rajesh', 7800, 'Finance');
109 ●  CALL InsertWorker(40, 'Kiran', 7200, 'Finance');
110 ●  CALL InsertWorker(41, 'Rakesh', 7500, 'Finance');
111 ●  CALL InsertWorker(42, 'Suresh', 7800, 'Finance');
112 ●  CALL InsertWorker(43, 'Rajesh', 7200, 'Finance');
113 ●  CALL InsertWorker(44, 'Kiran', 7500, 'Finance');
114 ●  CALL InsertWorker(45, 'Rakesh', 7800, 'Finance');
115 ●  CALL InsertWorker(46, 'Suresh', 7200, 'Finance');
116 ●  CALL InsertWorker(47, 'Rajesh', 7500, 'Finance');
117 ●  CALL InsertWorker(48, 'Kiran', 7800, 'Finance');
118 ●  CALL InsertWorker(49, 'Rakesh', 7200, 'Finance');
119 ●  CALL InsertWorker(50, 'Suresh', 7500, 'Finance');
120 ●  CALL InsertWorker(51, 'Rajesh', 7800, 'Finance');
121 ●  CALL InsertWorker(52, 'Kiran', 7200, 'Finance');
122 ●  CALL InsertWorker(53, 'Rakesh', 7500, 'Finance');
123 ●  CALL InsertWorker(54, 'Suresh', 7800, 'Finance');
124 ●  CALL InsertWorker(55, 'Rajesh', 7200, 'Finance');
125 ●  CALL InsertWorker(56, 'Kiran', 7500, 'Finance');
126 ●  CALL InsertWorker(57, 'Rakesh', 7800, 'Finance');
127 ●  CALL InsertWorker(58, 'Suresh', 7200, 'Finance');
128 ●  CALL InsertWorker(59, 'Rajesh', 7500, 'Finance');
129 ●  CALL InsertWorker(60, 'Kiran', 7800, 'Finance');
130 ●  CALL InsertWorker(61, 'Rakesh', 7200, 'Finance');
131 ●  CALL InsertWorker(62, 'Suresh', 7500, 'Finance');
132 ●  CALL InsertWorker(63, 'Rajesh', 7800, 'Finance');
133 ●  CALL InsertWorker(64, 'Kiran', 7200, 'Finance');
134 ●  CALL InsertWorker(65, 'Rakesh', 7500, 'Finance');
135 ●  CALL InsertWorker(66, 'Suresh', 7800, 'Finance');
136 ●  CALL InsertWorker(67, 'Rajesh', 7200, 'Finance');
137 ●  CALL InsertWorker(68, 'Kiran', 7500, 'Finance');
138 ●  CALL InsertWorker(69, 'Rakesh', 7800, 'Finance');
139 ●  CALL InsertWorker(70, 'Suresh', 7200, 'Finance');
140 ●  CALL InsertWorker(71, 'Rajesh', 7500, 'Finance');
141 ●  CALL InsertWorker(72, 'Kiran', 7800, 'Finance');
142 ●  CALL InsertWorker(73, 'Rakesh', 7200, 'Finance');
143 ●  CALL InsertWorker(74, 'Suresh', 7500, 'Finance');
144 ●  CALL InsertWorker(75, 'Rajesh', 7800, 'Finance');
145 ●  CALL InsertWorker(76, 'Kiran', 7200, 'Finance');
146 ●  CALL InsertWorker(77, 'Rakesh', 7500, 'Finance');
147 ●  CALL InsertWorker(78, 'Suresh', 7800, 'Finance');
148 ●  CALL InsertWorker(79, 'Rajesh', 7200, 'Finance');
149 ●  CALL InsertWorker(80, 'Kiran', 7500, 'Finance');
150 ●  CALL InsertWorker(81, 'Rakesh', 7800, 'Finance');
151 ●  CALL InsertWorker(82, 'Suresh', 7200, 'Finance');
152 ●  CALL InsertWorker(83, 'Rajesh', 7500, 'Finance');
153 ●  CALL InsertWorker(84, 'Kiran', 7800, 'Finance');
154 ●  CALL InsertWorker(85, 'Rakesh', 7200, 'Finance');
155 ●  CALL InsertWorker(86, 'Suresh', 7500, 'Finance');
156 ●  CALL InsertWorker(87, 'Rajesh', 7800, 'Finance');
157 ●  CALL InsertWorker(88, 'Kiran', 7200, 'Finance');
158 ●  CALL InsertWorker(89, 'Rakesh', 7500, 'Finance');
159 ●  CALL InsertWorker(90, 'Suresh', 7800, 'Finance');
160 ●  CALL InsertWorker(91, 'Rajesh', 7200, 'Finance');
161 ●  CALL InsertWorker(92, 'Kiran', 7500, 'Finance');
162 ●  CALL InsertWorker(93, 'Rakesh', 7800, 'Finance');
163 ●  CALL InsertWorker(94, 'Suresh', 7200, 'Finance');
164 ●  CALL InsertWorker(95, 'Rajesh', 7500, 'Finance');
165 ●  CALL InsertWorker(96, 'Kiran', 7800, 'Finance');
166 ●  CALL InsertWorker(97, 'Rakesh', 7200, 'Finance');
167 ●  CALL InsertWorker(98, 'Suresh', 7500, 'Finance');
168 ●  CALL InsertWorker(99, 'Rajesh', 7800, 'Finance');
169 ●  CALL InsertWorker(100, 'Kiran', 7200, 'Finance');
170 ●  CALL InsertWorker(101, 'Rakesh', 7500, 'Finance');
171 ●  CALL InsertWorker(102, 'Suresh', 7800, 'Finance');
172 ●  CALL InsertWorker(103, 'Rajesh', 7200, 'Finance');
173 ●  CALL InsertWorker(104, 'Kiran', 7500, 'Finance');
174 ●  CALL InsertWorker(105, 'Rakesh', 7800, 'Finance');
175 ●  CALL InsertWorker(106, 'Suresh', 7200, 'Finance');
176 ●  CALL InsertWorker(107, 'Rajesh', 7500, 'Finance');
177 ●  CALL InsertWorker(108, 'Kiran', 7800, 'Finance');
178 ●  CALL InsertWorker(109, 'Rakesh', 7200, 'Finance');
179 ●  CALL InsertWorker(110, 'Suresh', 7500, 'Finance');
180 ●  CALL InsertWorker(111, 'Rajesh', 7800, 'Finance');
181 ●  CALL InsertWorker(112, 'Kiran', 7200, 'Finance');
182 ●  CALL InsertWorker(113, 'Rakesh', 7500, 'Finance');
183 ●  CALL InsertWorker(114, 'Suresh', 7800, 'Finance');
184 ●  CALL InsertWorker(115, 'Rajesh', 7200, 'Finance');
185 ●  CALL InsertWorker(116, 'Kiran', 7500, 'Finance');
186 ●  CALL InsertWorker(117, 'Rakesh', 7800, 'Finance');
187 ●  CALL InsertWorker(118, 'Suresh', 7200, 'Finance');
188 ●  CALL InsertWorker(119, 'Rajesh', 7500, 'Finance');
189 ●  CALL InsertWorker(120, 'Kiran', 7800, 'Finance');
190 ●  CALL InsertWorker(121, 'Rakesh', 7200, 'Finance');
191 ●  CALL InsertWorker(122, 'Suresh', 7500, 'Finance');
192 ●  CALL InsertWorker(123, 'Rajesh', 7800, 'Finance');
193 ●  CALL InsertWorker(124, 'Kiran', 7200, 'Finance');
194 ●  CALL InsertWorker(125, 'Rakesh', 7500, 'Finance');
195 ●  CALL InsertWorker(126, 'Suresh', 7800, 'Finance');
196 ●  CALL InsertWorker(127, 'Rajesh', 7200, 'Finance');
197 ●  CALL InsertWorker(128, 'Kiran', 7500, 'Finance');
198 ●  CALL InsertWorker(129, 'Rakesh', 7800, 'Finance');
199 ●  CALL InsertWorker(130, 'Suresh', 7200, 'Finance');
200 ●  CALL InsertWorker(131, 'Rajesh', 7500, 'Finance');
201 ●  CALL InsertWorker(132, 'Kiran', 7800, 'Finance');
202 ●  CALL InsertWorker(133, 'Rakesh', 7200, 'Finance');
203 ●  CALL InsertWorker(134, 'Suresh', 7500, 'Finance');
204 ●  CALL InsertWorker(135, 'Rajesh', 7800, 'Finance');
205 ●  CALL InsertWorker(136, 'Kiran', 7200, 'Finance');
206 ●  CALL InsertWorker(137, 'Rakesh', 7500, 'Finance');
207 ●  CALL InsertWorker(138, 'Suresh', 7800, 'Finance');
208 ●  CALL InsertWorker(139, 'Rajesh', 7200, 'Finance');
209 ●  CALL InsertWorker(140, 'Kiran', 7500, 'Finance');
210 ●  CALL InsertWorker(141, 'Rakesh', 7800, 'Finance');
211 ●  CALL InsertWorker(142, 'Suresh', 7200, 'Finance');
212 ●  CALL InsertWorker(143, 'Rajesh', 7500, 'Finance');
213 ●  CALL InsertWorker(144, 'Kiran', 7800, 'Finance');
214 ●  CALL InsertWorker(145, 'Rakesh', 7200, 'Finance');
215 ●  CALL InsertWorker(146, 'Suresh', 7500, 'Finance');
216 ●  CALL InsertWorker(147, 'Rajesh', 7800, 'Finance');
217 ●  CALL InsertWorker(148, 'Kiran', 7200, 'Finance');
218 ●  CALL InsertWorker(149, 'Rakesh', 7500, 'Finance');
219 ●  CALL InsertWorker(150, 'Suresh', 7800, 'Finance');
220 ●  CALL InsertWorker(151, 'Rajesh', 7200, 'Finance');
221 ●  CALL InsertWorker(152, 'Kiran', 7500, 'Finance');
222 ●  CALL InsertWorker(153, 'Rakesh', 7800, 'Finance');
223 ●  CALL InsertWorker(154, 'Suresh', 7200, 'Finance');
224 ●  CALL InsertWorker(155, 'Rajesh', 7500, 'Finance');
225 ●  CALL InsertWorker(156, 'Kiran', 7800, 'Finance');
226 ●  CALL InsertWorker(157, 'Rakesh', 7200, 'Finance');
227 ●  CALL InsertWorker(158, 'Suresh', 7500, 'Finance');
228 ●  CALL InsertWorker(159, 'Rajesh', 7800, 'Finance');
229 ●  CALL InsertWorker(160, 'Kiran', 7200, 'Finance');
230 ●  CALL InsertWorker(161, 'Rakesh', 7500, 'Finance');
231 ●  CALL InsertWorker(162, 'Suresh', 7800, 'Finance');
232 ●  CALL InsertWorker(163, 'Rajesh', 7200, 'Finance');
233 ●  CALL InsertWorker(164, 'Kiran', 7500, 'Finance');
234 ●  CALL InsertWorker(165, 'Rakesh', 7800, 'Finance');
235 ●  CALL InsertWorker(166, 'Suresh', 7200, 'Finance');
236 ●  CALL InsertWorker(167, 'Rajesh', 7500, 'Finance');
237 ●  CALL InsertWorker(168, 'Kiran', 7800, 'Finance');
238 ●  CALL InsertWorker(169, 'Rakesh', 7200, 'Finance');
239 ●  CALL InsertWorker(170, 'Suresh', 7500, 'Finance');
240 ●  CALL InsertWorker(171, 'Rajesh', 7800, 'Finance');
241 ●  CALL InsertWorker(172, 'Kiran', 7200, 'Finance');
242 ●  CALL InsertWorker(173, 'Rakesh', 7500, 'Finance');
243 ●  CALL InsertWorker(174, 'Suresh', 7800, 'Finance');
244 ●  CALL InsertWorker(175, 'Rajesh', 7200, 'Finance');
245 ●  CALL InsertWorker(176, 'Kiran', 7500, 'Finance');
246 ●  CALL InsertWorker(177, 'Rakesh', 7800, 'Finance');
247 ●  CALL InsertWorker(178, 'Suresh', 7200, 'Finance');
248 ●  CALL InsertWorker(179, 'Rajesh', 7500, 'Finance');
249 ●  CALL InsertWorker(180, 'Kiran', 7800, 'Finance');
250 ●  CALL InsertWorker(181, 'Rakesh', 7200, 'Finance');
251 ●  CALL InsertWorker(182, 'Suresh', 7500, 'Finance');
252 ●  CALL InsertWorker(183, 'Rajesh', 7800, 'Finance');
253 ●  CALL InsertWorker(184, 'Kiran', 7200, 'Finance');
254 ●  CALL InsertWorker(185, 'Rakesh', 7500, 'Finance');
255 ●  CALL InsertWorker(186, 'Suresh', 7800, 'Finance');
256 ●  CALL InsertWorker(187, 'Rajesh', 7200, 'Finance');
257 ●  CALL InsertWorker(188, 'Kiran', 7500, 'Finance');
258 ●  CALL InsertWorker(189, 'Rakesh', 7800, 'Finance');
259 ●  CALL InsertWorker(190, 'Suresh', 7200, 'Finance');
260 ●  CALL InsertWorker(191, 'Rajesh', 7500, 'Finance');
261 ●  CALL InsertWorker(192, 'Kiran', 7800, 'Finance');
262 ●  CALL InsertWorker(193, 'Rakesh', 7200, 'Finance');
263 ●  CALL InsertWorker(194, 'Suresh', 7500, 'Finance');
264 ●  CALL InsertWorker(195, 'Rajesh', 7800, 'Finance');
265 ●  CALL InsertWorker(196, 'Kiran', 7200, 'Finance');
266 ●  CALL InsertWorker(197, 'Rakesh', 7500, 'Finance');
267 ●  CALL InsertWorker(198, 'Suresh', 7800, 'Finance');
268 ●  CALL InsertWorker(199, 'Rajesh', 7200, 'Finance');
269 ●  CALL InsertWorker(200, 'Kiran', 7500, 'Finance');
270 ●  CALL InsertWorker(201, 'Rakesh', 7800, 'Finance');
271 ●  CALL InsertWorker(202, 'Suresh', 7200, 'Finance');
272 ●  CALL InsertWorker(203, 'Rajesh', 7500, 'Finance');
273 ●  CALL InsertWorker(204, 'Kiran', 7800, 'Finance');
274 ●  CALL InsertWorker(205, 'Rakesh', 7200, 'Finance');
275 ●  CALL InsertWorker(206, 'Suresh', 7500, 'Finance');
276 ●  CALL InsertWorker(207, 'Rajesh', 7800, 'Finance');
277 ●  CALL InsertWorker(208, 'Kiran', 7200, 'Finance');
278 ●  CALL InsertWorker(209, 'Rakesh', 7500, 'Finance');
279 ●  CALL InsertWorker(210, 'Suresh', 7800, 'Finance');
280 ●  CALL InsertWorker(211, 'Rajesh', 7200, 'Finance');
281 ●  CALL InsertWorker(212, 'Kiran', 7500, 'Finance');
282 ●  CALL InsertWorker(213, 'Rakesh', 7800, 'Finance');
283 ●  CALL InsertWorker(214, 'Suresh', 7200, 'Finance');
284 ●  CALL InsertWorker(215, 'Rajesh', 7500, 'Finance');
285 ●  CALL InsertWorker(216, 'Kiran', 7800, 'Finance');
286 ●  CALL InsertWorker(217, 'Rakesh', 7200, 'Finance');
287 ●  CALL InsertWorker(218, 'Suresh', 7500, 'Finance');
288 ●  CALL InsertWorker(219, 'Rajesh', 7800, 'Finance');
289 ●  CALL InsertWorker(220, 'Kiran', 7200, 'Finance');
290 ●  CALL InsertWorker(221, 'Rakesh', 7500, 'Finance');
291 ●  CALL InsertWorker(222, 'Suresh', 7800, 'Finance');
292 ●  CALL InsertWorker(223, 'Rajesh', 7200, 'Finance');
293 ●  CALL InsertWorker(224, 'Kiran', 7500, 'Finance');
294 ●  CALL InsertWorker(225, 'Rakesh', 7800, 'Finance');
295 ●  CALL InsertWorker(226, 'Suresh', 7200, 'Finance');
296 ●  CALL InsertWorker(227, 'Rajesh', 7500, 'Finance');
297 ●  CALL InsertWorker(228, 'Kiran', 7800, 'Finance');
298 ●  CALL InsertWorker(229, 'Rakesh', 7200, 'Finance');
299 ●  CALL InsertWorker(230, 'Suresh', 7500, 'Finance');
300 ●  CALL InsertWorker(231, 'Rajesh', 7800, 'Finance');
301 ●  CALL InsertWorker(232, 'Kiran', 7200, 'Finance');
302 ●  CALL InsertWorker(233, 'Rakesh', 7500, 'Finance');
303 ●  CALL InsertWorker(234, 'Suresh', 7800, 'Finance');
304 ●  CALL InsertWorker(235, 'Rajesh', 7200, 'Finance');
305 ●  CALL InsertWorker(236, 'Kiran', 7500, 'Finance');
306 ●  CALL InsertWorker(237, 'Rakesh', 7800, 'Finance');
307 ●  CALL InsertWorker(238, 'Suresh', 7200, 'Finance');
308 ●  CALL InsertWorker(239, 'Rajesh', 7500, 'Finance');
309 ●  CALL InsertWorker(240, 'Kiran', 7800, 'Finance');
310 ●  CALL InsertWorker(241, 'Rakesh', 7200, 'Finance');
311 ●  CALL InsertWorker(242, 'Suresh', 7500, 'Finance');
312 ●  CALL InsertWorker(243, 'Rajesh', 7800, 'Finance');
313 ●  CALL InsertWorker(244, 'Kiran', 7200, 'Finance');
314 ●  CALL InsertWorker(245, 'Rakesh', 7500, 'Finance');
315 ●  CALL InsertWorker(246, 'Suresh', 7800, 'Finance');
316 ●  CALL InsertWorker(247, 'Rajesh', 7200, 'Finance');
317 ●  CALL InsertWorker(248, 'Kiran', 7500, 'Finance');
318 ●  CALL InsertWorker(249, 'Rakesh', 7800, 'Finance');
319 ●  CALL InsertWorker(250, 'Suresh', 7200, 'Finance');
320 ●  CALL InsertWorker(251, 'Rajesh', 7500, 'Finance');
321 ●  CALL InsertWorker(252, 'Kiran', 7800, 'Finance');
322 ●  CALL InsertWorker(253, 'Rakesh', 7200, 'Finance');
323 ●  CALL InsertWorker(254, 'Suresh', 7500, 'Finance');
324 ●  CALL InsertWorker(255, 'Rajesh', 7800, 'Finance');
325 ●  CALL InsertWorker(256, 'Kiran', 7200, 'Finance');
326 ●  CALL InsertWorker(257, 'Rakesh', 7500, 'Finance');
327 ●  CALL InsertWorker(258, 'Suresh', 7800, 'Finance');
328 ●  CALL InsertWorker(259, 'Rajesh', 7200, 'Finance');
329 ●  CALL InsertWorker(260, 'Kiran', 7500, 'Finance');
330 ●  CALL InsertWorker(261, 'Rakesh', 7800, 'Finance');
331 ●  CALL InsertWorker(262, 'Suresh', 7200, 'Finance');
332 ●  CALL InsertWorker(263, 'Rajesh', 7500, 'Finance');
333 ●  CALL InsertWorker(264, 'Kiran', 7800, 'Finance');
334 ●  CALL InsertWorker(265, 'Rakesh', 7200, 'Finance');
335 ●  CALL InsertWorker(266, 'Suresh', 7500, 'Finance');
336 ●  CALL InsertWorker(267, 'Rajesh', 7800, 'Finance');
337 ●  CALL InsertWorker(268, 'Kiran', 7200, 'Finance');
338 ●  CALL InsertWorker(269, 'Rakesh', 7500, 'Finance');
339 ●  CALL InsertWorker(270, 'Suresh', 7800, 'Finance');
340 ●  CALL InsertWorker(271, 'Rajesh', 7200, 'Finance');
341 ●  CALL InsertWorker(272, 'Kiran', 7500, 'Finance');
342 ●  CALL InsertWorker(273, 'Rakesh', 7800, 'Finance');
343 ●  CALL InsertWorker(274, 'Suresh', 7200, 'Finance');
344 ●  CALL InsertWorker(275, 'Rajesh', 7500, 'Finance');
345 ●  CALL InsertWorker(276, 'Kiran', 7800, 'Finance');
346 ●  CALL InsertWorker(277, 'Rakesh', 7200, 'Finance');
347 ●  CALL InsertWorker(278, 'Suresh', 7500, 'Finance');
348 ●  CALL InsertWorker(279, 'Rajesh', 7800, 'Finance');
349 ●  CALL InsertWorker(280, 'Kiran', 7200, 'Finance');
350 ●  CALL InsertWorker(281, 'Rakesh', 7500, 'Finance');
351 ●  CALL InsertWorker(282, 'Suresh', 7800, 'Finance');
352 ●  CALL InsertWorker(283, 'Rajesh', 7200, 'Finance');
353 ●  CALL InsertWorker(284, 'Kiran', 7500, 'Finance');
354 ●  CALL InsertWorker(285, 'Rakesh', 7800, 'Finance');
355 ●  CALL InsertWorker(286, 'Suresh', 7200, 'Finance');
356 ●  CALL InsertWorker(287, 'Rajesh', 7500, 'Finance');
357 ●  CALL InsertWorker(288, 'Kiran', 7800, 'Finance');
358 ●  CALL InsertWorker(289, 'Rakesh', 7200, 'Finance');
359 ●  CALL InsertWorker(290, 'Suresh', 7500, 'Finance');
360 ●  CALL InsertWorker(291, 'Rajesh', 7800, 'Finance');
361 ●  CALL InsertWorker(292, 'Kiran', 7200, 'Finance');
362 ●  CALL InsertWorker(293, 'Rakesh', 7500, 'Finance');
363 ●  CALL InsertWorker(294, 'Suresh', 7800, 'Finance');
364 ●  CALL InsertWorker(295, 'Rajesh', 7200, 'Finance');
365 ●  CALL InsertWorker(296, 'Kiran', 7500, 'Finance');
366 ●  CALL InsertWorker(297, 'Rakesh', 7800, 'Finance');
367 ●  CALL InsertWorker(298, 'Suresh', 7200, 'Finance');
368 ●  CALL InsertWorker(299, 'Rajesh', 7500, 'Finance');
369 ●  CALL InsertWorker(300, 'Kiran', 7800, 'Finance');
370 ●  CALL InsertWorker(301, 'Rakesh', 7200, 'Finance');
371 ●  CALL InsertWorker(302, 'Suresh', 7500, 'Finance');
372 ●  CALL InsertWorker(303, 'Rajesh', 7800, 'Finance');
373 ●  CALL InsertWorker(304, 'Kiran', 7200, 'Finance');
374 ●  CALL InsertWorker(305, 'Rakesh', 7500, 'Finance');
375 ●  CALL InsertWorker(306, 'Suresh', 7800, 'Finance');
376 ●  CALL InsertWorker(307, 'Rajesh', 7200, 'Finance');
377 ●  CALL InsertWorker(308, 'Kiran', 7500, 'Finance');
378 ●  CALL InsertWorker(309, 'Rakesh', 7800, 'Finance');
379 ●  CALL InsertWorker(310, 'Suresh', 7200, 'Finance');
380 ●  CALL InsertWorker(311, 'Rajesh', 7500, 'Finance');
381 ●  CALL InsertWorker(312, 'Kiran', 7800, 'Finance');
382 ●  CALL InsertWorker(313, 'Rakesh', 7200, 'Finance');
383 ●  CALL InsertWorker(314, 'Suresh', 7500, 'Finance');
384 ●  CALL InsertWorker(315, 'Rajesh', 7800, 'Finance');
385 ●  CALL InsertWorker(316, 'Kiran', 7200, 'Finance');
386 ●  CALL InsertWorker(317, 'Rakesh', 7500, 'Finance');
387 ●  CALL InsertWorker(318, 'Suresh', 7800, 'Finance');
388 ●  CALL InsertWorker(319, 'Rajesh', 7200, 'Finance');
389 ●  CALL InsertWorker(320, 'Kiran', 7500, 'Finance');
390 ●  CALL InsertWorker(321, 'Rakesh', 7800, 'Finance');
391 ●  CALL InsertWorker(322, 'Suresh', 7200, 'Finance');
392 ●  CALL InsertWorker(323, 'Rajesh', 7500, 'Finance');
393 ●  CALL InsertWorker(324, 'Kiran', 7800, 'Finance');
394 ●  CALL InsertWorker(325, 'Rakesh', 7200, 'Finance');
395 ●  CALL InsertWorker(326, 'Suresh', 7500, 'Finance');
396 ●  CALL InsertWorker(327, 'Rajesh', 7800, 'Finance');
397 ●  CALL InsertWorker(328, 'Kiran', 7200, 'Finance');
398 ●  CALL InsertWorker(329, 'Rakesh', 7500, 'Finance');
399 ●  CALL InsertWorker(330, 'Suresh', 7800, 'Finance');
400 ●  CALL InsertWorker(331, 'Rajesh', 7200, 'Finance');
401 ●  CALL InsertWorker(332, 'Kiran', 7500, 'Finance');
402 ●  CALL InsertWorker(333, 'Rakesh', 7800, 'Finance');
403 ●  CALL InsertWorker(334, 'Suresh', 7200, 'Finance');
404 ●  CALL InsertWorker(335, 'Rajesh', 7500, 'Finance');
405 ●  CALL InsertWorker(336, 'Kiran', 7800, 'Finance');
406 ●  CALL InsertWorker(337, 'Rakesh', 7200, 'Finance');
407 ●  CALL InsertWorker(338, 'Suresh', 7500, 'Finance');
408 ●  CALL InsertWorker(339, 'Rajesh', 7800, 'Finance');
409 ●  CALL InsertWorker(340, 'Kiran', 7200, 'Finance');
410 ●  CALL InsertWorker(341, 'Rakesh', 7500, 'Finance');
411 ●  CALL InsertWorker(342, 'Suresh', 7800, 'Finance');
412 ●  CALL InsertWorker(343, 'Rajesh', 7200, 'Finance');
413 ●  CALL InsertWorker(344, 'Kiran', 7500, 'Finance');
414 ●  CALL InsertWorker(345, 'Rakesh', 7800, 'Finance');
415 ●  CALL InsertWorker(346, 'Suresh', 7200, 'Finance');
416 ●  CALL InsertWorker(347, 'Rajesh', 7500, 'Finance');
417 ●  CALL InsertWorker(348, 'Kiran', 7800, 'Finance');
418 ●  CALL InsertWorker(349, 'Rakesh', 7200, 'Finance');
419 ●  CALL InsertWorker(350, 'Suresh', 7500, 'Finance');
420 ●  CALL InsertWorker(351, 'Rajesh', 7800, 'Finance');
421 ●  CALL InsertWorker(352, 'Kiran', 7200, 'Finance');
422 ●  CALL InsertWorker(353, 'Rakesh', 7500, 'Finance');
423 ●  CALL InsertWorker(354, 'Suresh', 7800, 'Finance');
424 ●  CALL InsertWorker(355, 'Rajesh', 7200, 'Finance');
425 ●  CALL InsertWorker(356, 'Kiran', 7500, 'Finance');
426 ●  CALL InsertWorker(357, 'Rakesh', 7800, 'Finance');
427 ●  CALL InsertWorker(358, 'Suresh', 7200, 'Finance');
428 ●  CALL InsertWorker(359, 'Rajesh', 7500, 'Finance');
429 ●  CALL InsertWorker(360, 'Kiran', 7800, 'Finance');
430 ●  CALL InsertWorker(361, 'Rakesh', 7200, 'Finance');
431 ●  CALL InsertWorker(362, 'Suresh', 7500, 'Finance');
432 ●  CALL InsertWorker(363, 'Rajesh', 7800, 'Finance');
433 ●  CALL InsertWorker(364, 'Kiran', 7200, 'Finance');
434 ●  CALL InsertWorker(365, 'Rakesh', 7500, 'Finance');
435 ●  CALL InsertWorker(366, 'Suresh', 7800, 'Finance');
436 ●  CALL InsertWorker(367, 'Rajesh', 7200, 'Finance');
437 ●  CALL InsertWorker(368, 'Kiran', 7500, 'Finance');
438 ●  CALL InsertWorker(369, 'Rakesh', 7800, '
```

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

```

78 -- Write stored procedure takes in an IN parameter for WORKER_ID and an OUT parameter for SALARY.
79 -- 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.
80
81 DELIMITER $$ 
82 • CREATE PROCEDURE GetSalary(
83   IN p_Worker_Id INT,
84   OUT p_Salary INT
85 )
86 BEGIN
87   SELECT Salary INTO p_Salary FROM Worker WHERE Worker_Id=p_Worker_ID;
88 END $$ 
89 DELIMITER ;
90 • CALL GetSalary(1, @Salary);
91 • SELECT @Salary;
92
93
94
95
96
97

```

Output

#	Time	Action	Message	Duration / Fetch
10	23:19:53	CALL InsertWorker(5, 'Raj', 7800, 'Finance')	1 row(s) affected	0.015 sec
11	23:20:00	SELECT * FROM Worker LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
12	23:34:08	CREATE PROCEDURE GetSalary(IN p_Worker_Id INT, OUT p_Salary INT) BEGIN SELECT Salary INTO ...	0 row(s) affected	0.000 sec
13	23:34:37	SELECT * FROM Worker LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
14	23:35:55	SELECT * FROM Worker LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
15	23:38:10	CALL GetSalary(1, @Salary)	1 row(s) affected	0.016 sec
16	23:38:35	SELECT @Salary LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

11:38 PM
2/2/2024

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.

```

92
93 -- Create a stored procedure that takes in IN parameters for WORKER_ID and DEPARTMENT.
94 -- It should update the department of the worker with the given ID. Then make a procedure call.
95 DELIMITER $$ 
96 • CREATE PROCEDURE Update_dept(
97   IN p_Worker_Id INT,
98   IN p_Department VARCHAR(30)
99 )
100 BEGIN
101   UPDATE Worker SET DEPARTMENT = p_Department WHERE Worker_Id=p_Worker_Id;
102 END $$ 
103 DELIMITER ;
104 • CALL Update_dept(2, 'New_department');
105 • SELECT * FROM Worker;
106
107
108
109
110
111

```

Output

#	Time	Action	Message	Duration / Fetch
1	12:44:31	CREATE PROCEDURE Update_dept(IN p_Worker_Id INT, IN p_Department VARCHAR(30)) BEGIN UP...	0 row(s) affected	0.000 sec
2	12:44:40	USE PRODUCT	0 row(s) affected	0.000 sec
3	12:44:51	CREATE PROCEDURE Update_dept(IN p_Worker_Id INT, IN p_Department VARCHAR(30)) BEGIN UP...	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

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.

```

107  -- Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_workerCount.
108  -- It should retrieve the number of workers in the given department and returns it in the p_workerCount parameter. Make procedure call.
109  DELIMITER $$ 
110  CREATE PROCEDURE worker_count(
111      IN p_Department VARCHAR(30),
112      OUT p_workerCount INT
113  )
114  BEGIN
115      SELECT COUNT(*) INTO p_WorkerCount FROM Worker WHERE Department=p_Department;
116  END $$ 
117  DELIMITER ;
118  CALL Worker_count('Analytics', @workercount);
119  SELECT @workercount;
120
121
122
123
124
125
126

```

Action	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) BEG...	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	

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.

Action	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	

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.

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code pane contains the following SQL:

```
-- 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.
DELIMITER $$

CREATE TRIGGER before_insert_teacher
BEFORE INSERT ON Teacher
FOR EACH ROW
BEGIN
    IF NEW.Salary < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Salary cannot be negative';
    END IF;
END;
$$
DELIMITER ;

SHOW TRIGGERS;
UPDATE Teacher SET Salary=55000 WHERE Id=1;
SELECT * FROM Teacher;
-- This will trigger the error message
INSERT INTO teacher VALUES
(10, 'Mark Black', 'Music', 3, -3000.00);

```

The output pane shows the results of the executed statements:

Action	Time	Message	Duration / Fetch
SELECT * FROM Teacher LIMIT 0, 1000	11 22:15:50	8 row(s) returned	0.000 sec / 0.000 sec
CREATE TRIGGER before_insert_teacher BEFORE INSERT ON Teacher FOR EACH ROW BEGIN IF ... END IF; END;	12 22:15:59	0 row(s) affected	0.000 sec
SHOW TRIGGERS	13 22:16:18	1 row(s) returned	0.016 sec / 0.000 sec
UPDATE Teacher SET Salary=-12000 WHERE Id=1	14 22:18:17	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
SELECT * FROM Teacher LIMIT 0, 1000	15 22:18:36	8 row(s) returned	0.000 sec / 0.000 sec
INSERT INTO teacher VALUES (10, 'Mark Black', 'Music', 3, -3000.00)	16 22:18:54	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
INSERT INTO teacher VALUES (10, 'Mark Black', 'Music', 3, -3000.00)	17 22:19:58	Error Code: 1644. Salary cannot be negative	0.000 sec

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.

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Vigilator SQL File 10* SQL File 3* SQL Clauses* Bank data* SQL Functions* SQL VIEW* Joins* Views* Triggers* x

HEMAS Filter objects

bank constraints func joins orgdb product sales student sys teachers Tables teacher Columns Indexes ministration Schemas formation

Table: teacher

Columns:

- id** int PK
- name** varchar(30)
- subject** varchar(30)
- experience** int
- salary** decimal(7,2)

```

76 • CREATE TABLE teacher_log (log_id INT PRIMARY KEY AUTO_INCREMENT, teacher_id INT, action VARCHAR(10), timestamp TIMESTAMP);
77
78 • DELIMITER $$;
79 • CREATE TRIGGER after_insert_teacher
80 AFTER INSERT ON teacher
81 FOR EACH ROW
82 BEGIN
83     -- Insert into teacher_log table
84     INSERT INTO teacher_log (teacher_id, action, timestamp)
85     VALUES (NEW.id, 'INSERT', CURRENT_TIMESTAMP);
86 END;
87 $$;
88
89 DELIMITER ;
90
91 • INSERT INTO Teacher VALUES (12, 'Mike', 'Maths', 12, 7500.00);
92 • INSERT INTO Teacher VALUES (14, 'Josh', 'Maths', 12, 7500.00);
93 • SELECT * FROM Teacher_log;
94

```

Action Output

#	Time	Action	Message	Duration / Fetch
18	13:40:49	DROP TRIGGER IF EXISTS after_insert_teacher	0 row(s) affected	0.000 sec
19	13:41:15	CREATE TABLE teacher_log (log_id INT PRIMARY KEY AUTO_INCREMENT, teacher_id INT, action VARCHAR(10), timestamp TIMESTAMP);	0 row(s) affected	0.016 sec
20	13:41:25	CREATE TRIGGER after_insert_teacher AFTER INSERT ON teacher FOR EACH ROW BEGIN -- Insert int...	0 row(s) affected	0.000 sec
21	13:42:43	INSERT INTO Teacher VALUES (12, 'Mike', 'Maths', 12, 7500.00)	1 row(s) affected	0.000 sec
22	13:42:50	INSERT INTO Teacher VALUES (12, 'Mike', 'Maths', 12, 7500.00)	Error Code: 1062: Duplicate entry '12' for key teacher.PRIMARY'	0.000 sec
23	13:43:18	INSERT INTO Teacher VALUES (14, 'Josh', 'Maths', 12, 7500.00)	1 row(s) affected	0.016 sec
24	13:43:39	SELECT * FROM Teacher_log LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

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.

Information

Table: teacher

Columns:

- id** int PK
- name** varchar(30)
- subject** varchar(30)
- experience** int
- salary** decimal(7,2)

```

94
95 -- Create a before delete trigger that will raise an error when you try to delete a row that has experience greater than 10 years.
96 DELIMITER $$;
97 • CREATE TRIGGER before_delete_teacher
98 BEFORE DELETE ON teacher
99 FOR EACH ROW
100 BEGIN
101 IF OLD.experience > 10 THEN
102     SIGNAL SQLSTATE '45000'
103     SET MESSAGE_TEXT = 'Cannot delete a teacher with experience greater than 10 years';
104 END IF;
105 END;
106 $$;
107 DELIMITER ;
108 • DELETE FROM Teacher WHERE Id=14;
109
110

```

Action Output

#	Time	Action	Message	Duration / Fetch
20	13:41:25	CREATE TRIGGER after_insert_teacher AFTER INSERT ON teacher FOR EACH ROW BEGIN -- Insert int...	0 row(s) affected	0.000 sec
21	13:42:43	INSERT INTO Teacher VALUES (12, 'Mike', 'Maths', 12, 7500.00)	1 row(s) affected	0.000 sec
22	13:42:50	INSERT INTO Teacher VALUES (12, 'Mike', 'Maths', 12, 7500.00)	Error Code: 1062: Duplicate entry '12' for key teacher.PRIMARY'	0.000 sec
23	13:43:18	INSERT INTO Teacher VALUES (14, 'Josh', 'Maths', 12, 7500.00)	1 row(s) affected	0.016 sec
24	13:43:39	SELECT * FROM Teacher_log LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
25	17:21:34	CREATE TRIGGER before_delete_teacher BEFORE DELETE ON teacher FOR EACH ROW BEGIN IF OLD...	0 row(s) affected	0.000 sec
26	17:22:28	DELETE FROM Teacher WHERE Id=14	Error Code: 1544: Cannot delete a teacher with experience greater than 10 years	0.015 sec

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

5:41 PM 2/7/2024