

NAME: Divyansh Kumar Singh

SAP ID: 590012175

BATCH: B1

DBMS LAB 3

AIM: To implement and use SQL Sub-Query

Lab Objective: To understand the use of SQL Subquery.

Divyansh Kumar Singh
590012175

classmate

Date 26/8/25
Page _____

W LAB-3

26/8
60

CREATE

DESC

FROM

WHERE

IN

INSERT INTO

SELECT

ANY

SET

DELETE

MAX, MIN

COUNT

Avg

GROUP BY

AS

EXISTS

Row

FLOOR

⇒ 46 (Approx)

MySQL Subquery

A subquery in MySQL is nested into another SQL query and embedded with SELECT, INSERT, UPDATE, or DELETE statements and the various operators. We can also nest the subquery with another subquery. A subquery is known as the **inner query**, and the query that contains a subquery is known as the **outer query**. The inner query executed first gives the result to the outer query, and then the main/outer query will be performed. MySQL allows us to use subquery anywhere but must be closed within parenthesis. All subquery forms and operations supported by the SQL standard will also be supported in MySQL.

The following are the rules to use subqueries:

- Subqueries should always be in **parentheses**.
- If the main query does not have multiple columns for the subquery, then a subquery can have only one column in the SELECT command.
- We can use various comparison operators with the subquery, such as **>**, **<**, **=**, **IN**, **ANY**, **SOME**, and **ALL**.

A multiple-row operator is very useful when the subquery returns more than one row.

- We cannot use the **ORDER BY** clause in a subquery, although it can be used inside the main query.
- If we use a subquery in a **set function**, it cannot be immediately enclosed in a set function.

The following are the advantages of using subqueries:

- The subqueries make the queries in a structured form that allows us to isolate each part of a statement.

- The subqueries provide alternative ways to query the data from the table; otherwise, we need to use complex joins and unions.
- The subqueries are more readable than complex join or union statements.

SQL Subquery

The Subquery or Inner query is an SQL query placed inside another SQL query. It is embedded in the HAVING or WHERE clause of the SQL statements.

Following are the important rules which the SQL Subquery must follow:

1. The SQL subqueries can be used with the following statements along with the SQL expression operators:
 - SELECT statement,
 - UPDATE statement,
 - INSERT statement, and
 - DELETE statement.
2. The subqueries in SQL are always enclosed in the parenthesis and placed on the right side of the SQL operators.
3. We cannot use the ORDER BY clause in the subquery. But, we can use the GROUP BY clause, which performs the same function as the ORDER BY clause.
4. If the subquery returns more than one record, we must use the multiple value operators before the subquery.
5. We can use the BETWEEN operator within the subquery but not with the subquery.

Subquery with SELECT statement

In SQL, the SELECT statement uses inner or nested queries most frequently. The syntax of the subquery with the SELECT statement is described in the following block:

```
SELECT Column_Name1, Column_Name2, ...., Column_NameN  
FROM Table_Name WHERE Column_Name Comparison_Operator  
( SELECT Column_Name1, Column_Name2, ...., Column_NameN  
FROM Table_Name WHERE condition);
```

Subquery with the INSERT statement

We can also use the subqueries and nested queries with the INSERT statement in Structured Query Language. We can insert the subquery results into the table of the outer query. The syntax of the subquery with the INSERT statement is described in the following block:

```
INSERT INTO Table_Name SELECT * FROM Table_Name WHERE  
Column_Name Operator (Subquery);
```

Subquery with the UPDATE statement

The subqueries and nested queries can be used with the UPDATE statement in Structured Query Language to update the existing table's columns. We can easily update one or more columns using a subquery with the UPDATE statement.

Syntax of Subquery with the UPDATE statement

```
UPDATE Table_Name SET Column_Name = New_value WHERE Value  
OPERATOR (SELECT COLUMN_NAME FROM TABLE_NAME  
WHERE Condition) ;
```

Subquery with the DELETE statement

We can easily delete one or more records from the SQL table using subquery with the DELETE statement in Structured Query Language.

Syntax of Subquery with DELETE statement

```
DELETE FROM Table_Name WHERE Value OPERATOR (SELECT  
COLUMN_NAME FROM TABLE_NAME WHERE Condition);
```

MySQL Subquery with Comparison Operator

A comparison operator is an operator used to compare values and returns the result, either true or false. The following comparison operators are used in MySQL <, >, =, <>, <=>, etc. We can use the subquery before or after the comparison operators that return a single value. The returned value can be the arithmetic expression or a column function. After that, SQL compares the subquery results with the value on the other side of the comparison operator.

The example below explains it more clearly:

Following is a simple statement that returns the **employee detail whose income is more than 350000** with the help of a subquery:

```
SELECT * FROM employees  
WHERE emp_id IN (SELECT emp_id FROM employees  
WHERE income > 350000);
```

This query first executes the subquery that returns the **employee id whose income > 350000**. Second, the main query will return the employees all details whose employee id are in the result set returned by the subquery. After executing the statement, we will get the below output, where we can see the employee detail whose income>350000.

Let us see an example of another comparison operator, such as equality (=) to find employee details with **maximum income** using a subquery.

```
SELECT emp_name, city, income FROM employees  
WHERE income = (SELECT MAX(income) FROM employees);
```

It will give the output where we can see two employees detail who have maximum income.

MySQL Subquery with IN or NOT-IN Operator

If the subquery produces more than one value, we need to use the IN or NOT IN Operator with the WHERE clause.

Suppose we have a table named "Student" and "Student2" that contains the following data:

Table: Student

Stud_ID	Name	Email	City
1	Peter	peter@javatpoint.com	Texas
2	Suzi	suzi@javatpoint.com	California
3	Joseph	joseph@javatpoint.com	Alaska
4	Andrew	andrew@javatpoint.com	Los Angeles
5	Brayan	brayan@javatpoint.com	New York

Table: Student2

Stud_ID	Name	Email	City
1	Stephen	stephen@javatpoint.com	Texas
2	Joseph	joseph@javatpoint.com	Los Angeles
3	Peter	peter@javatpoint.com	California
4	David	david@javatpoint.com	New York
5	Maddy	maddy@javatpoint.com	Los Angeles

The following subquery with NOT IN Operator returns the **student detail who does not belong to Los Angeles City** from both tables as follows:

```
SELECT Name, City FROM student
WHERE City NOT IN (
SELECT City FROM student2 WHERE City='Los Angeles');
```

After execution, we can see that the result contains the student details that do not belong to Los Angeles City.

MySQL Subquery in the FROM Clause

If we use a subquery in the FROM clause, MySQL will return the output from a subquery is used as a temporary table. We called this table as a derived table, inline views, or materialized subquery. The following subquery returns the maximum, minimum, and average number of items in the order table:

```
SELECT Max(items), MIN(items), FLOOR(AVG(items))
FROM
(SELECT order_id, COUNT(order_id) AS items FROM orders
GROUP BY order_date) AS Student_order_detail;
```

MySQL Correlated Subqueries

A correlated subquery in MySQL is a subquery that depends on the outer query. It uses the data from the outer query or contains a reference to a parent query that also appears in the outer query. MySQL evaluates it once from each row in the outer query.

```
SELECT emp_name, city, income
FROM employees emp WHERE income > (
SELECT AVG(income) FROM employees WHERE city = emp.city);
```

In the above query, we select an **employee name and city** whose income is higher than the average income of all employees in each city. The subquery executes for every city of the specified table because it will change for every row.

Therefore, the average income will also be changed. Then, the main query filters employee details whose income is higher than the average income from the subquery.

MySQL Subqueries with EXISTS or NOT EXISTS

The EXISTS operator is a Boolean operator that returns a true or false result. It is used with a subquery and checks the existence of data in a subquery. If a subquery returns any record at all, this Operator returns true. Otherwise,

it will return false. The NOT EXISTS Operator used for negation that gives true value when the subquery does not return any row. Otherwise, it returns false. Both EXISTS and NOT EXISTS used with correlated subqueries. The following example illustrates it more clearly. Suppose we have a table **customer and order**.

The below SQL statements uses EXISTS operator to find the name, occupation, and age of the customer who has placed at least one order.

```
SELECT name, occupation, age FROM customer C
WHERE EXISTS (SELECT * FROM Orders O
WHERE C.cust_id = O.cust_id);
```

This statement uses NOT EXISTS operator that returns the customer details who have not placed an order.

```
SELECT name, occupation, age FROM customer C
WHERE NOT EXISTS (SELECT * FROM Orders O
WHERE C.cust_id = O.cust_id);
```

We can see the below output to understand the above queries result.

MySQL ROW Subqueries

It is a subquery that returns a single row where we can get more than one column values. We can use the following operators for comparing row subqueries =, >, <, >=, <=, \diamond , !=, \neq . Let us see the following example:

```
SELECT * FROM customer C WHERE ROW(cust_id, occupation) = (
SELECT order_id, order_date FROM Orders O WHERE C.cust_id =
O.cust_id);
```

If given row has cust_id, occupation values equal to the order_id, order_date values of any rows in the first table, the WHERE expression is TRUE, and each query returns those first table rows. Otherwise, the expression is FALSE, and the query produces an empty set.

MySQL Subqueries with ALL, ANY, and SOME

We can use a subquery which is followed by the keyword ALL, ANY, or SOME after a comparison operator. The following are the syntax to use subqueries with ALL, ANY, or SOME:

operand comparison_operator ANY (subquery)
operand comparison_operator ALL (subquery)
operand comparison_operator SOME (subquery)

The ALL keyword compares values with the value returned by a subquery. Therefore, it returns TRUE if the comparison is TRUE for ALL of the values returned by a subquery.

The ANY keyword returns TRUE if the comparison is TRUE for ANY of the values returned by a subquery.

The ANY and SOME keywords are the same because they are the alias of each other. The following example explains it more clearly:

```
SELECT cust_id, name FROM customer WHERE
cust_id > ANY (SELECT cust_id FROM Orders);
```

If we use ALL in place of ANY, it will return TRUE when the comparison is TRUE for ALL values in the column returned by a subquery. For example:

```
SELECT cust_id, name FROM customer WHERE
cust_id > ALL (SELECT cust_id FROM Orders);
```

Lab Performance Questions

Tables:

1. Student_Details

- Columns: Student_RollNo, Stu_Name, Stu_Marks, Stu_City

The screenshot shows the MySQL Workbench interface. In the Schemas pane, a database named 'Lab_3' is selected. In the central SQL editor, the following SQL code is displayed:

```
1 • CREATE DATABASE Lab_3;
2 USE Lab_3;
3
4 -- 1. Student_Details
5 • ⊖ CREATE TABLE Student_Details (
6     Student_RollNo INT PRIMARY KEY,
7     Stu_Name VARCHAR(50),
8     Stu_Marks INT,
9     Stu_City VARCHAR(50)
10 );
11 • DESC Student_Details;
```

The Result Grid pane shows the structure of the 'Student_Details' table:

Field	Type	Null	Key	Default	Extra
Student_RollNo	int	NO	PRI	NULL	
Stu_Name	varchar(50)	YES		NULL	
Stu_Marks	int	YES		NULL	
Stu_City	varchar(50)	YES		NULL	

The Action Output pane displays the execution log:

	Time	Action	Response	Duration / Fetch Time
✓ 1	21:25:54	CREATE DATABASE...	1 row(s) affected	0.0015 sec
✓ 2	21:25:54	USE Lab_3	0 row(s) affected	0.00024 sec
✓ 3	21:26:01	CREATE TABLE S...	0 row(s) affected	0.0053 sec
✓ 4	21:26:01	DESC Student_D...	4 row(s) returned	0.0018 sec / 0.00001...

2. Faculty_Details

- Columns: Faculty_ID, Name, Dept_ID, Address

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

- Schemas:** The current schema is "Lab_3".
- SQL Editor:** The code entered is:

```
DESC Student_Details;
12
13 -- 2. Faculty_Details
14 * CREATE TABLE Faculty_Details (
15     Faculty_ID INT PRIMARY KEY,
16     Name VARCHAR(50),
17     Dept_ID INT,
18     Address VARCHAR(100)
19 );
20 * DESC Faculty_Details;
21
```
- Result Grid:** A table showing the structure of the Faculty_Details table:

Field	Type	Null	Key	Default	Extra
Faculty_ID	int	NO	PRI	NULL	
Name	varchar(50)	YES		NULL	
Dept_ID	int	YES		NULL	
Address	varchar(100)	YES		NULL	
- Action Output:** A table showing the execution history:

Time	Action	Response	Duration / Fetch Time
21:25:54	CREATE DATABASE	1 row(s) affected	0.0015 sec
21:25:54	USE Lab_3	0 row(s) affected	0.00024 sec
21:26:01	CREATE TABLE S...	0 row(s) affected	0.0053 sec
21:26:01	DESC Student_D...	4 row(s) returned	0.0018 sec / 0.00001...
21:30:37	CREATE TABLE F...	0 row(s) affected	0.0076 sec
21:30:37	DESC Faculty_De...	4 row(s) returned	0.0021 sec / 0.00001...

3. Department

- Columns: Dept_ID, Faculty_ID, Dept_Name

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

- Schemas:** The current schema is "Lab_3".
- SQL Editor:** The query window contains the following SQL code:

```
DESC Faculty_Details;
21
22 -- 3. Department
23 * CREATE TABLE Department (
24     Dept_ID INT PRIMARY KEY,
25     Faculty_ID INT,
26     Dept_Name VARCHAR(50),
27     FOREIGN KEY (Faculty_ID) REFERENCES Faculty_Details(Faculty_ID)
28 );
29 * DESC Department;
30
```
- Result Grid:** A table showing the structure of the Department table:

Field	Type	Null	Key	Default	Extra
Dept_ID	int	NO	PRI	NULL	
Faculty_ID	int	YES	MUL	NULL	
Dept_Name	varchar(50)	YES		NULL	
- Action Output:** A table showing the history of database actions:

	Time	Action	Response	Duration / Fetch Time
✓ 1	21:25:54	CREATE DATABASE...	1 row(s) affected	0.0015 sec
✓ 2	21:25:54	USE Lab_3	0 row(s) affected	0.00024 sec
✓ 3	21:26:01	CREATE TABLE S...	0 row(s) affected	0.0053 sec
✓ 4	21:26:01	DESC Student_D...	4 row(s) returned	0.0018 sec / 0.00001...
✓ 5	21:30:37	CREATE TABLE F...	0 row(s) affected	0.0076 sec
✓ 6	21:30:37	DESC Faculty_De...	4 row(s) returned	0.0021 sec / 0.00001...
✓ 7	21:31:23	CREATE TABLE D...	0 row(s) affected	0.0095 sec
✓ 8	21:31:23	DESC Department	3 row(s) returned	0.0027 sec / 0.00001...

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

SCHEMAS Filter objects

- > Basic
- > Bus_Station
- Lab_3
 - > Tables
 - Views
 - Stored Pr...
 - Functions
- > Sample
- > sys
- > TechCo

29 • DESC Department;
30
31 -- 3b. Department_Emp
32 • CREATE TABLE Department_Emp (
33 Dept_ID INT PRIMARY KEY,
34 Dept_Name VARCHAR(50),
35 Emp_ID INT,
36 Dept_Grade CHAR(1)
37);
38 • DESC Department_Emp;
39

100% 21:38

Result Grid Filter Rows: Search Export: Result Grid

Field	Type	Null	Key	Default	Extra
Dept_ID	int	NO	PRI	NULL	
Dept_Name	varchar(50)	YES		NULL	
Emp_ID	int	YES		NULL	
Dept_Grade	char(1)	YES		NULL	

Result 4 Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
✓ 1	21:25:54	CREATE DATABASE	1 row(s) affected	0.0015 sec
✓ 2	21:25:54	USE Lab_3	0 row(s) affected	0.00024 sec
✓ 3	21:26:01	CREATE TABLE S...	0 row(s) affected	0.0053 sec
✓ 4	21:26:01	DESC Student_D...	4 row(s) returned	0.0018 sec / 0.00001...
✓ 5	21:30:37	CREATE TABLE F...	0 row(s) affected	0.0076 sec
✓ 6	21:30:37	DESC Faculty_De...	4 row(s) returned	0.0021 sec / 0.00001...
✓ 7	21:31:23	CREATE TABLE D...	0 row(s) affected	0.0095 sec
✓ 8	21:31:23	DESC Department	3 row(s) returned	0.0027 sec / 0.00001...
✓ 9	21:31:44	CREATE TABLE D...	0 row(s) affected	0.018 sec
✓ 10	21:31:44	DESC Departmen...	4 row(s) returned	0.0083 sec / 0.00001...

4. Old_Employee

- Columns: Emp_ID, Emp_Name, Emp_Salary, Address

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

- Title Bar:** MySQL Workbench, Divyansh Kumar Singh 590012175
- Toolbar:** Standard MySQL Workbench icons for file operations, schema navigation, and database management.
- Schemas Tab:** Shows the current schema is "Lab_3".
- SQL Editor:** Contains the SQL code for creating the "Old_Employee" table and its description.

```
39
40      -- 4. Old_Employee
41  •  CREATE TABLE Old_Employee (
42      Emp_ID INT PRIMARY KEY,
43      Emp_Name VARCHAR(50),
44      Emp_Salary DECIMAL(10,2),
45      Address VARCHAR(100)
46  );
47  •  DESC Old_Employee;
48
```
- Result Grid:** Displays the structure of the "Old_Employee" table with four columns: Emp_ID, Emp_Name, Emp_Salary, and Address.

Field	Type	Null	Key	Default	Extra
Emp_ID	int	NO	PRI	NULL	
Emp_Name	varchar(50)	YES		NULL	
Emp_Salary	decimal(10,2)	YES		NULL	
Address	varchar(100)	YES		NULL	

- Action Output:** Shows the history of database actions with their times, descriptions, responses, and durations.

	Time	Action	Response	Duration / Fetch Time
✓ 3	21:26:01	CREATE TABLE S...	0 row(s) affected	0.0053 sec
✓ 4	21:26:01	DESC Student_D...	4 row(s) returned	0.0018 sec / 0.00001...
✓ 5	21:30:37	CREATE TABLE F...	0 row(s) affected	0.0076 sec
✓ 6	21:30:37	DESC Faculty_De...	4 row(s) returned	0.0021 sec / 0.00001...
✓ 7	21:31:23	CREATE TABLE D...	0 row(s) affected	0.0095 sec
✓ 8	21:31:23	DESC Department	3 row(s) returned	0.0027 sec / 0.00001...
✓ 9	21:31:44	CREATE TABLE D...	0 row(s) affected	0.018 sec
✓ 10	21:31:44	DESC Departmen...	4 row(s) returned	0.0083 sec / 0.00001...
✓ 11	21:32:41	CREATE TABLE O...	0 row(s) affected	0.021 sec
✓ 12	21:32:41	DESC Old_Emplo...	4 row(s) returned	0.0029 sec / 0.00002...

- Status Bar:** Query Completed

5. New_Employee

- Columns: Emp_ID, Emp_Name, Emp_Salary, Address

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

- Schemas:** The current schema is "Lab_3".
- SQL Editor:** Contains the following SQL code:

```
47 * DESC Old_Employee;
48 *
49 * -- 5. New_Employee
50 * CREATE TABLE New_Employee (
51 *     Emp_ID INT PRIMARY KEY,
52 *     Emp_Name VARCHAR(50),
53 *     Emp_Salary DECIMAL(10,2),
54 *     Address VARCHAR(100)
55 * );
56 * DESC New_Employee;
```
- Result Grid:** Displays the structure of the New_Employee table:

Field	Type	Null	Key	Default	Extra
Emp_ID	int	NO	PRI	NULL	
Emp_Name	varchar(50)	YES		NULL	
Emp_Salary	decimal(10,2)	YES		NULL	
Address	varchar(100)	YES		NULL	
- Action Output:** Shows the execution history of the queries:

Time	Action	Response	Duration / Fetch Time
21:30:37	CREATE TABLE Faculty_De...	0 row(s) affected	0.00/6 sec
21:30:37	DESC Faculty_De...	4 row(s) returned	0.0021 sec / 0.00001...
21:31:23	CREATE TABLE Department...	0 row(s) affected	0.0095 sec
21:31:23	DESC Department...	3 row(s) returned	0.0027 sec / 0.00001...
21:31:44	CREATE TABLE Old_Employee...	0 row(s) affected	0.018 sec
21:31:44	DESC Old_Employee...	4 row(s) returned	0.0083 sec / 0.00001...
21:32:41	CREATE TABLE Old_Employee...	0 row(s) affected	0.021 sec
21:32:41	DESC Old_Employee...	4 row(s) returned	0.0029 sec / 0.00002...
21:33:53	CREATE TABLE New_Employee...	0 row(s) affected	0.012 sec
21:33:53	DESC New_Employee...	4 row(s) returned	0.0028 sec / 0.00002...
- Status:** "Query Completed"

6. Employee_Details

- Columns: Emp_ID, Emp_Name, Emp_Salary, Dept_ID

The screenshot shows the MySQL Workbench interface. In the left sidebar under 'SCHEMAS', the 'Lab_3' schema is selected. The main pane displays the SQL code for creating the 'Employee_Details' table:

```
-- 6. Employee_Details
CREATE TABLE Employee_Details (
    Emp_ID INT PRIMARY KEY,
    Emp_Name VARCHAR(50),
    Emp_Salary DECIMAL(10,2),
    Dept_ID INT,
    FOREIGN KEY (Dept_ID) REFERENCES Department_Emp(Dept_ID)
);
DESC Employee_Details;
```

The 'Result Grid' tab shows the table structure with four columns: Emp_ID, Emp_Name, Emp_Salary, and Dept_ID. The 'Action Output' tab at the bottom lists the execution history of the queries.

Field	Type	Null	Key	Default	Extra
Emp_ID	int	NO	PRI	NULL	
Emp_Name	varchar(50)	YES		NULL	
Emp_Salary	decimal(10,2)	YES		NULL	
Dept_ID	int	YES	MUL	NULL	

Time	Action	Response	Duration / Fetch Time
7 21:31:23	CREATE TABLE D...	0 row(s) affected	0.0095 sec
8 21:31:23	DESC Department	3 row(s) returned	0.0027 sec / 0.00001...
9 21:31:44	CREATE TABLE D...	0 row(s) affected	0.018 sec
10 21:31:44	DESC Departmen...	4 row(s) returned	0.0083 sec / 0.00001...
11 21:32:41	CREATE TABLE O...	0 row(s) affected	0.021 sec
12 21:32:41	DESC Old_Emplo...	4 row(s) returned	0.0029 sec / 0.00002...
13 21:33:53	CREATE TABLE N...	0 row(s) affected	0.012 sec
14 21:33:53	DESC New_Empl...	4 row(s) returned	0.0028 sec / 0.00002...
15 21:34:37	CREATE TABLE E...	0 row(s) affected	0.012 sec
16 21:34:37	DESC Employee_...	4 row(s) returned	0.0037 sec / 0.00002...

Query Completed

7. Student

- Columns: Student_ID, Name, City

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

- Schemas:** The current schema is "Lab_3".
- Query Editor:** The query being run is:

```
DESC Employee_Details;
-- 7. Student
CREATE TABLE Student (
    Student_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    City VARCHAR(50)
);
DESC Student;
```
- Result Grid:** The table structure for "Student" is displayed:

Field	Type	Null	Key	Default	Extra
Student_ID	int	NO	PRI	HULL	
Name	varchar(50)	YES		HULL	
City	varchar(50)	YES		HULL	
- Action Output:** A log of database actions is shown:

Time	Action	Response	Duration / Fetch Time
21:31:44	CREATE TABLE D...	0 row(s) affected	0.018 sec
21:31:44	DESC Departmen...	4 row(s) returned	0.0083 sec / 0.00001...
21:32:41	CREATE TABLE O...	0 row(s) affected	0.021 sec
21:32:41	DESC Old_Emplo...	4 row(s) returned	0.0029 sec / 0.00002...
21:33:53	CREATE TABLE N...	0 row(s) affected	0.012 sec
21:33:53	DESC New_Empl...	4 row(s) returned	0.0028 sec / 0.00002...
21:34:37	CREATE TABLE E...	0 row(s) affected	0.012 sec
21:34:37	DESC Employee_...	4 row(s) returned	0.0037 sec / 0.00002...
21:35:19	CREATE TABLE S...	0 row(s) affected	0.026 sec
21:35:19	DESC Student	3 row(s) returned	0.0055 sec / 0.00001...
- Status:** The message "Query Completed" is at the bottom.

8. Student2

- Columns: Student_ID, City

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

- Schemas:** The current schema is "Lab_3".
- SQL Editor:** The query being run is:

```
-- 8. Student2
CREATE TABLE Student2 (
    Student_ID INT PRIMARY KEY,
    City VARCHAR(50)
);
DESC Student;
```

- Result Grid:** The results of the DESCRIBE command for the Student2 table.

Field	Type	Null	Key	Default	Extra
Student_ID	int	NO	PRI	NUL	
City	varchar(50)	YES		NUL	

- Action Output:** A table showing the history of actions taken in the session.

Time	Action	Response	Duration / Fetch Time
21:32:41	CREATE TABLE O...	0 row(s) affected	0.021 sec
21:32:41	DESC Old_Emplo...	4 row(s) returned	0.0029 sec / 0.00002...
21:33:53	CREATE TABLE N...	0 row(s) affected	0.012 sec
21:33:53	DESC New_Empl...	4 row(s) returned	0.0028 sec / 0.00002...
21:34:37	CREATE TABLE E...	0 row(s) affected	0.012 sec
21:34:37	DESC Employee_...	4 row(s) returned	0.0037 sec / 0.00002...
21:35:19	CREATE TABLE S...	0 row(s) affected	0.026 sec
21:35:19	DESC Student	3 row(s) returned	0.0055 sec / 0.00001...
21:36:24	CREATE TABLE S...	0 row(s) affected	0.0047 sec
21:36:24	DESC Student2	2 row(s) returned	0.0012 sec / 0.00000...

- Status:** Query Completed

9. Orders

- Columns: Order_ID, Cust_ID, Order_Date

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

- Schemas:** The current schema is "Lab_3".
- Query Editor:** The query being run is:

```
DESC Student2;
82
83 -- 9. Orders
84 • ⊖ CREATE TABLE Orders (
85     Order_ID INT PRIMARY KEY,
86     Cust_ID INT,
87     Order_Date DATE
88 );
89 • DESC Orders;
90
```
- Result Grid:** The table structure for the Orders table is displayed:

Field	Type	Null	Key	Default	Extra
Order_ID	int	NO	PRI	NULL	
Cust_ID	int	YES		NULL	
Order_Date	date	YES		NULL	
- Action Output:** A table showing the history of actions taken on the database, including the creation of tables and descriptions of them. The last few rows are:

Time	Action	Response	Duration / Fetch Time
13 21:33:53	CREATE TABLE N...	0 row(s) affected	0.012 sec
14 21:33:53	DESC New_Empl...	4 row(s) returned	0.0028 sec / 0.00002...
15 21:34:37	CREATE TABLE E...	0 row(s) affected	0.012 sec
16 21:34:37	DESC Employee_...	4 row(s) returned	0.0037 sec / 0.00002...
17 21:35:19	CREATE TABLE S...	0 row(s) affected	0.026 sec
18 21:35:19	DESC Student	3 row(s) returned	0.0055 sec / 0.00001...
19 21:36:24	CREATE TABLE S...	0 row(s) affected	0.0047 sec
20 21:36:24	DESC Student2	2 row(s) returned	0.0012 sec / 0.0000...
21 21:36:58	CREATE TABLE O...	0 row(s) affected	0.0066 sec
22 21:36:58	DESC Orders	3 row(s) returned	0.0014 sec / 0.00001...
- Status:** The status bar at the bottom left says "Query Completed".

10. Sales

- Columns: Product_Category, Sales_Amount

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

- Schemas:** The current schema is "Lab_3".
- Query Editor:** The query being run is:

```
);  
DESC Orders;  
90  
91 -- 10. Sales  
92 * CREATE TABLE Sales (  
93     Product_Category VARCHAR(50),  
94     Sales_Amount DECIMAL(10,2)  
95 );  
96 * DESC Sales;  
97
```

- Result Grid:** The table structure for "Sales" is displayed.

Field	Type	Null	Key	Default	Extra
Product_Category	varchar(50)	YES		NULL	
Sales_Amount	decimal(10,2)	YES		NULL	

- Action Output:** A table showing the history of actions taken:

Time	Action	Response	Duration / Fetch Time
15 21:34:37	CREATE TABLE Employee	0 row(s) affected	0.012 sec
16 21:34:37	DESC Employee	4 row(s) returned	0.0037 sec / 0.00002...
17 21:35:19	CREATE TABLE Student	0 row(s) affected	0.026 sec
18 21:35:19	DESC Student	3 row(s) returned	0.0055 sec / 0.00001...
19 21:36:24	CREATE TABLE Student2	0 row(s) affected	0.0047 sec
20 21:36:24	DESC Student2	2 row(s) returned	0.0012 sec / 0.00000...
21 21:36:58	CREATE TABLE Orders	0 row(s) affected	0.0066 sec
22 21:36:58	DESC Orders	3 row(s) returned	0.0014 sec / 0.00001...
23 21:37:29	CREATE TABLE Sales	0 row(s) affected	0.0063 sec
24 21:37:29	DESC Sales	2 row(s) returned	0.0013 sec / 0.00000...

- Status:** Query Completed.

11. Top_Students

- Columns: Student_ID, Top_Marks

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

- Schemas:** The current schema is "Lab_3".
- Code Editor:** The code being run is:

```
95    );
96 • DESC Sales;
97
98 -- 11. Top_Students
99 • CREATE TABLE Top_Students (
100     Student_ID INT PRIMARY KEY,
101     Top_Marks INT
102 );
103 • DESC Top_Students;
```

- Result Grid:** The table structure is displayed:

Field	Type	Null	Key	Default	Extra
Student_ID	int	NO	PRI	NULL	
Top_Marks	int	YES		NULL	

- Action Output:** A log of database actions is shown:

Time	Action	Response	Duration / Fetch Time
21:35:19	CREATE TABLE S...	0 row(s) affected	0.026 sec
21:35:19	DESC Student	3 row(s) returned	0.0055 sec / 0.00001...
21:36:24	CREATE TABLE S...	0 row(s) affected	0.0047 sec
21:36:24	DESC Student2	2 row(s) returned	0.0012 sec / 0.00000...
21:36:58	CREATE TABLE O...	0 row(s) affected	0.0066 sec
21:36:58	DESC Orders	3 row(s) returned	0.0014 sec / 0.00001...
21:37:29	CREATE TABLE S...	0 row(s) affected	0.0063 sec
21:37:29	DESC Sales	2 row(s) returned	0.0013 sec / 0.00000...
21:38:00	CREATE TABLE T...	0 row(s) affected	0.0063 sec
21:38:00	DESC Top_Stude...	2 row(s) returned	0.0014 sec / 0.00000...

- Status:** The message "Query Completed" is at the bottom.

Customer Table because its required for few subquery question.

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

- Schemas:** The current schema is "Lab_3".
- Code Editor:** The code being run is:

```
104
105      -- 12. Customer
106 • CREATE TABLE Customer (
107      Cust_ID INT PRIMARY KEY,
108      Name VARCHAR(50),
109      Age INT,
110      Occupation VARCHAR(50)
111 );
112 • DESC Customer;
```

- Result Grid:** Shows the structure of the Customer table with four columns: Cust_ID, Name, Age, and Occupation.

Field	Type	Null	Key	Default	Extra
Cust_ID	int	NO	PRI	NULL	
Name	varchar(50)	YES		NULL	
Age	int	YES		NULL	
Occupation	varchar(50)	YES		NULL	

- Action Output:** A table showing the history of actions with columns: ID, Time, Action, Response, and Duration / Fetch Time.

ID	Time	Action	Response	Duration / Fetch Time
19	21:36:24	CREATE TABLE S...	0 row(s) affected	0.004 sec
20	21:36:24	DESC Student2	2 row(s) returned	0.0012 sec / 0.00000...
21	21:36:58	CREATE TABLE O...	0 row(s) affected	0.0066 sec
22	21:36:58	DESC Orders	3 row(s) returned	0.0014 sec / 0.00001...
23	21:37:29	CREATE TABLE S...	0 row(s) affected	0.0063 sec
24	21:37:29	DESC Sales	2 row(s) returned	0.0013 sec / 0.00000...
25	21:38:00	CREATE TABLE T...	0 row(s) affected	0.0063 sec
26	21:38:00	DESC Top_Stude...	2 row(s) returned	0.0014 sec / 0.00000...
27	21:38:58	CREATE TABLE C...	0 row(s) affected	0.0063 sec
28	21:38:58	DESC Customer	4 row(s) returned	0.0016 sec / 0.00000...

- Status:** Query Completed.

Inserting data into the tables created:

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

- Title Bar:** MySQL Workbench
- Toolbar:** Standard MySQL Workbench toolbar with icons for Home, Schemas, Tables, Views, Functions, Procedures, Triggers, Events, and Utilities.
- Schemas Tab:** Shows the current schema is "Lab_3".
- SQL Editor:** Displays the following SQL code:

```
117
118 • INSERT INTO Student_Details VALUES
119     (1001, 'Akhil', 85, 'Agra'),
120     (1002, 'Balram', 78, 'Delhi'),
121     (1003, 'Bheem', 87, 'Gurgaon'),
122     (1004, 'Chetan', 95, 'Noida'),
123     (1005, 'Diksha', 99, 'Agra'),
124     (1006, 'Raman', 90, 'Ghaziabad'),
125     (1007, 'Sheetal', 68, 'Delhi);
126 • SELECT * FROM Student_Details;
```
- Result Grid:** Shows the results of the SELECT query:

Student_RollNo	Stu_Name	Stu_Marks	Stu_City
1001	Akhil	85	Agra
1002	Balram	78	Delhi
1003	Bheem	87	Gurgaon
1004	Chetan	95	Noida
1005	Diksha	99	Agra
1006	Raman	90	Ghaziabad
1007	Sheetal	68	Delhi
- Action Output:** Shows the history of database actions with their times, descriptions, responses, and durations.

Time	Action	Response	Duration / Fetch Time
21:38:00	CREATE TABLE ...	0 row(s) affected	0.0003 sec
21:38:00	DESC Top_Stude...	2 row(s) returned	0.0014 sec / 0.00000...
21:38:58	CREATE TABLE C...	0 row(s) affected	0.0063 sec
21:38:58	DESC Customer	4 row(s) returned	0.0016 sec / 0.00000...
21:41:15	INSERT INTO Stu...	7 row(s) affected Records: 7 Duplicates: 0...	0.0019 sec
21:41:15	SELECT * FROM...	7 row(s) returned	0.00048 sec / 0.000...
- Status:** Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Filter objects

127

128 • INSERT INTO Faculty_Details VALUES
(101, 'Bheem', 1, 'Gurgaon'),
(102, 'Chetan', 2, 'Noida'),
(103, 'Diksha', NULL, 'Agra'),
(104, 'Raman', 4, 'Ghaziabad'),
(105, 'Yatin', 3, 'Noida'),
(106, 'Anuj', NULL, 'Agra'),
(107, 'Rakes', 5, 'Gurgaon');

136 • SELECT * FROM Faculty_Details;

137

100% 31:136

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Faculty_ID	Name	Dept_ID	Address
101	Bheem	1	Gurgaon
102	Chetan	2	Noida
103	Diksha	NULL	Agra
104	Raman	4	Ghaziabad
105	Yatin	3	Noida
106	Anuj	NULL	Agra
107	Rakes	5	Gurgaon

Faculty_Details 15

Action Output

Time	Action	Response	Duration / Fetch Time
27 21:30:00	CREATE TABLE C...	0 row(s) affected	0.0003 sec
28 21:38:58	DESC Customer	4 row(s) returned	0.0016 sec / 0.00000...
29 21:41:15	INSERT INTO Stu...	7 row(s) affected Records: 7 Duplicates: 0...	0.0019 sec
30 21:41:15	SELECT * FROM...	7 row(s) returned	0.00048 sec / 0.000...
31 21:42:16	INSERT INTO Fac...	7 row(s) affected Records: 7 Duplicates: 0...	0.0015 sec
32 21:42:16	SELECT * FROM...	7 row(s) returned	0.00029 sec / 0.0000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

Filter objects

(106, 'Anuj', NULL, 'Agra'),
(107, 'Rakes', 5, 'Gurgaon');
SELECT * FROM Faculty_Details;

137
138 • INSERT INTO Department VALUES
139 (1, 101, 'BCA'),
140 (2, 102, 'B.Tech'),
141 (3, 105, 'BBA'),
142 (4, 104, 'MBA'),
143 (5, 107, 'MCA');
144 • SELECT * FROM Department;

100% 26:144

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Dept_ID	Faculty_ID	Dept_Name
1	101	BCA
2	102	B.Tech
3	105	BBA
4	104	MBA
5	107	MCA
NULL	NULL	NULL

Department 16 Apply

Action Output

Time	Action	Response	Duration / Fetch Time
29 21:41:10	INSERT INTO Stu...	7 Row(s) affected Records: 7 Duplicates: 0...	0.0019 sec
30 21:41:15	SELECT * FROM...	7 row(s) returned	0.00048 sec / 0.000...
31 21:42:16	INSERT INTO Fac...	7 row(s) affected Records: 7 Duplicates: 0...	0.0015 sec
32 21:42:16	SELECT * FROM...	7 row(s) returned	0.00029 sec / 0.0000...
33 21:43:12	INSERT INTO De...	5 row(s) affected Records: 5 Duplicates: 0...	0.0011 sec
34 21:43:12	SELECT * FROM...	5 row(s) returned	0.00046 sec / 0.000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Filter objects

Basic
Bus_Station
Lab_3
Tables
Views
Stored Pr...
Functions
Sample
sys
TechCo

Sample 2 | Divyansh Kumar Singh 590012175 Lab 3* | Lab3 (1) | >

Limit to 1000 rows

145
146 • INSERT INTO Department_Emp VALUES
147 (401, 'Administration', 1008, 'C'),
148 (402, 'HR', 1004, 'A'),
149 (403, 'Testing', 1002, 'C'),
150 (404, 'Coding', 1001, 'B'),
151 (405, 'Sales', 1003, 'A'),
152 (406, 'Marketing', NULL, 'C'),
153 (407, 'Accounting', 1005, 'C');
154 • SELECT * FROM Department_Emp;
155

100% 30:154

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Dept_ID	Dept_Name	Emp_ID	Dept_Grade
401	Administration	1008	C
402	HR	1004	A
403	Testing	1002	C
404	Coding	1001	B
405	Sales	1003	A
406	Marketing	NULL	C
407	Accounting	1005	C
NULL	NULL	NULL	NULL

Department_Emp 17 Apply

Action Output

Time	Action	Response	Duration / Fetch Time
21:42:10	INSERT INTO De...	7 row(s) affected Records: 7 Duplicates: 0...	0.0010 sec
21:42:16	SELECT * FROM...	7 row(s) returned	0.00029 sec / 0.0000...
21:43:12	INSERT INTO De...	5 row(s) affected Records: 5 Duplicates: 0...	0.0011 sec
21:43:12	SELECT * FROM...	5 row(s) returned	0.00046 sec / 0.000...
21:45:02	INSERT INTO De...	7 row(s) affected Records: 7 Duplicates: 0...	0.0024 sec
21:45:02	SELECT * FROM...	7 row(s) returned	0.00038 sec / 0.000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

SCHEMAS Filter objects

> Basic
> Bus_Station
Lab_3
Tables
Views
Stored Pr...
Functions
> Sample
> sys
> TechCo

155
156 • INSERT INTO Old_Employee VALUES
157 (1001, 'Akhil', 50000, 'Agra'),
158 (1002, 'Balram', 25000, 'Delhi'),
159 (1003, 'Bheem', 45000, 'Gurgaon'),
160 (1004, 'Chetan', 60000, 'Noida'),
161 (1005, 'Diksha', 30000, 'Agra'),
162 (1006, 'Raman', 50000, 'Ghaziabad'),
163 (1007, 'Sheetal', 35000, 'Delhi');
164 • SELECT * FROM Old_Employee;
165

100% 28:164

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

	Emp_ID	Emp_Name	Emp_Salary	Address
	1001	Akhil	50000.00	Agra
	1002	Balram	25000.00	Delhi
	1003	Bheem	45000.00	Gurgaon
	1004	Chetan	60000.00	Noida
	1005	Diksha	30000.00	Agra
	1006	Raman	50000.00	Ghaziabad
	1007	Sheetal	35000.00	Delhi
	NULL	NULL	NULL	NULL

Old_Employee 18 Apply

Action Output

	Time	Action	Response	Duration / Fetch Time
33	21:43:12	INSERT INTO De...	5 row(s) affected Records: 5 Duplicates: 0...	0.0011 sec
34	21:43:12	SELECT * FROM...	5 row(s) returned	0.00046 sec / 0.000...
35	21:45:02	INSERT INTO De...	7 row(s) affected Records: 7 Duplicates: 0...	0.0024 sec
36	21:45:02	SELECT * FROM...	7 row(s) returned	0.00038 sec / 0.000...
37	21:45:48	INSERT INTO Old...	7 row(s) affected Records: 7 Duplicates: 0...	0.0019 sec
38	21:45:48	SELECT * FROM...	7 row(s) returned	0.00040 sec / 0.000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

SCHEMAS Filter objects

- > Basic
- > Bus_Station
- < Lab_3
 - > Tables
 - > Views
 - > Stored Pr...
 - > Functions
- > Sample
- > sys
- > TechCo

(1005, 'Diksha', 30000, 'Agra'),
(1006, 'Raman', 50000, 'Ghaziabad'),
163 (1007, 'Sheetal', 35000, 'Delhi');
164 • SELECT * FROM Old_Employee;
165
166 • INSERT INTO New_Employee VALUES
167 (1008, 'Sumit', 50000, 'Agra'),
168 (1009, 'Akash', 55000, 'Delhi'),
169 (1010, 'Devansh', 65000, 'Gurgaon');
170 • SELECT * FROM New_Employee;
171

100% | 28:170

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid

	Emp_ID	Emp_Name	Emp_Salary	Address
	1008	Sumit	50000.00	Agra
	1009	Akash	55000.00	Delhi
	1010	Devansh	65000.00	Gurgaon
	HULL	HULL	HULL	HULL

New_Employee 19

Action Output

	Time	Action	Response	Duration / Fetch Time
35	21:45:02	INSERT INTO De...	7 row(s) affected Records: 7 Duplicates: 0...	0.0024 sec
36	21:45:02	SELECT * FROM...	7 row(s) returned	0.00038 sec / 0.000...
37	21:45:48	INSERT INTO Old...	7 row(s) affected Records: 7 Duplicates: 0...	0.0019 sec
38	21:45:48	SELECT * FROM...	7 row(s) returned	0.00040 sec / 0.000...
39	21:46:36	INSERT INTO Ne...	3 row(s) affected Records: 3 Duplicates: 0...	0.0018 sec
40	21:46:36	SELECT * FROM...	3 row(s) returned	0.00030 sec / 0.000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

SCHEMAS Filter objects

- > Basic
- > Bus_Station
- < Lab_3
 - > Tables
 - Views
 - Stored Pr...
 - Functions
- > Sample
- > sys
- > TechCo

171
172 * INSERT INTO Employee_Details VALUES
173 (1001, 'Akhil', 50000, 404),
174 (1002, 'Balram', 25000, 403),
175 (1003, 'Bheem', 45000, 405),
176 (1004, 'Chetan', 60000, 402),
177 (1005, 'Ram', 65000, 407),
178 (1006, 'Shyam', 55500, 401),
179 (1007, 'Shobhit', 60000, 406);
180 * SELECT * FROM Employee_Details;
181

100% 32:180

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000.00	404
1002	Balram	25000.00	403
1003	Bheem	45000.00	405
1004	Chetan	60000.00	402
1005	Ram	65000.00	407
1006	Shyam	55500.00	401
1007	Shobhit	60000.00	406
NULL	NULL	NULL	NULL

Employee_Details 20 Apply

Action Output

Time	Action	Response	Duration / Fetch Time
21:45:48	INSERT INTO Empl...	7 row(s) affected Records: 7 Duplicates: 0...	0.00015 sec
21:45:48	SELECT * FROM...	7 row(s) returned	0.00040 sec / 0.000...
21:46:36	INSERT INTO Ne...	3 row(s) affected Records: 3 Duplicates: 0...	0.0018 sec
21:46:36	SELECT * FROM...	3 row(s) returned	0.00030 sec / 0.000...
21:47:19	INSERT INTO Em...	7 row(s) affected Records: 7 Duplicates: 0...	0.0025 sec
21:47:19	SELECT * FROM...	7 row(s) returned	0.00051 sec / 0.000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Filter objects

Basic Bus_Station Lab_3 Sample sys TechCo

Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) Limit to 1000 rows

(1007, 'Shobhit', 60000, 406);
180 • SELECT * FROM Employee_Details;
181
182 • INSERT INTO Student VALUES
183 (1, 'Peter', 'Texas'),
184 (2, 'Suzi', 'California'),
185 (3, 'Joseph', 'Alaska'),
186 (4, 'Andrew', 'Los Angeles'),
187 (5, 'Brayan', 'New York');
188 • SELECT * FROM Student;
189

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Student_ID	Name	City
1	Peter	Texas
2	Suzi	California
3	Joseph	Alaska
4	Andrew	Los Angeles
5	Brayan	New York
HULL	HULL	HULL

Action Output

Time	Action	Response	Duration / Fetch Time
39 21:46:30	INSERT INTO Ne...	3 row(s) affected Records: 3 Duplicates: 0...	0.0010 sec
40 21:46:36	SELECT * FROM...	3 row(s) returned	0.00030 sec / 0.000...
41 21:47:19	INSERT INTO Em...	7 row(s) affected Records: 7 Duplicates: 0...	0.0025 sec
42 21:47:19	SELECT * FROM...	7 row(s) returned	0.00051 sec / 0.0000...
43 21:47:50	INSERT INTO Stu...	5 row(s) affected Records: 5 Duplicates: 0...	0.0018 sec
44 21:47:50	SELECT * FROM...	5 row(s) returned	0.00028 sec / 0.0000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Q Filter objects

> Basic
> Bus_Station
Lab_3
> Tables
Views
Stored Pr...
Functions
> Sample
> sys
> TechCo

Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

(5, 'Brayan', 'New York');
188 • SELECT * FROM Student;
189
190 • INSERT INTO Student2 VALUES
191 (1, 'Texas'),
192 (2, 'California'),
193 (3, 'Alaska'),
194 (4, 'Los Angeles'),
195 (5, 'New York');
196 • SELECT * FROM Student2;
197
100% 24:196

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid

	Student_ID	City
1	Texas	
2	California	
3	Alaska	
4	Los Angeles	
5	New York	
HULL	HULL	

Student2 22 Apply

Action Output

	Time	Action	Response	Duration / Fetch Time
41	21:47:19	INSERT INTO Stu...	7 row(s) affected Records: 7 Duplicates: 0...	0.0023 sec
42	21:47:19	SELECT * FROM...	7 row(s) returned	0.00051 sec / 0.0000...
43	21:47:50	INSERT INTO Stu...	5 row(s) affected Records: 5 Duplicates: 0...	0.0018 sec
44	21:47:50	SELECT * FROM...	5 row(s) returned	0.00028 sec / 0.0000...
45	21:48:24	INSERT INTO Stu...	5 row(s) affected Records: 5 Duplicates: 0...	0.0020 sec
46	21:48:24	SELECT * FROM...	5 row(s) returned	0.00035 sec / 0.0000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

SCHEMAS Filter objects

- > Basic
- > Bus_Station
- < Lab_3
 - > Tables
 - Views
 - Stored Pr...
 - Functions
- > Sample
- > sys
- > TechCo

(5, 'New York');

```
196 •   SELECT * FROM Student2;
197
198 •   INSERT INTO Orders VALUES
199     (1, 101, '2023-01-15'),
200     (2, 102, '2023-02-20'),
201     (3, 103, '2023-03-10'),
202     (4, 104, '2023-04-05'),
203     (5, 105, '2023-05-12');
204 •   SELECT * FROM Orders;
205
```

100% 22:204

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Order_ID	Cust_ID	Order_Date
1	101	2023-01-15
2	102	2023-02-20
3	103	2023-03-10
4	104	2023-04-05
5	105	2023-05-12
NONE	NONE	NONE

Orders 23 Apply

Action Output

Time	Action	Response	Duration / Fetch Time
21:47:00	INSERT INTO Stu...	5 row(s) affected Records: 5 Duplicates: 0...	0.0018 sec
21:47:50	SELECT * FROM...	5 row(s) returned	0.00028 sec / 0.0000...
21:48:24	INSERT INTO Stu...	5 row(s) affected Records: 5 Duplicates: 0...	0.0020 sec
21:48:24	SELECT * FROM...	5 row(s) returned	0.00035 sec / 0.0000...
21:48:53	INSERT INTO Ord...	5 row(s) affected Records: 5 Duplicates: 0...	0.0021 sec
21:48:53	SELECT * FROM...	5 row(s) returned	0.00042 sec / 0.0000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Filter objects

> Basic
> Bus_Station
Lab_3
Tables
Views
Stored Pr...
Functions
> Sample
> sys
> TechCo

Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

Limit to 1000 rows

(4, 104, '2023-04-05'),
203 (5, 105, '2023-05-12');
204 • SELECT * FROM Orders;
205
206 • INSERT INTO Sales VALUES
207 ('Electronics', 15000.00),
208 ('Clothing', 8500.00),
209 ('Books', 3200.00),
210 ('Furniture', 21000.00);
211 • SELECT * FROM Sales;
212

100% 1:212

Result Grid Filter Rows: Search Export:

Product_Categ...	Sales_Amount
Electronics	15000.00
Clothing	8500.00
Books	3200.00
Furniture	21000.00

Sales 24

Result Grid Form Editor

Read Only

Action Output

Time	Action	Response	Duration / Fetch Time
21:48:24	INSERT INTO Sales	5 row(s) affected Records: 5 Duplicates: 0...	0.00020 sec
46 21:48:24	SELECT * FROM...	5 row(s) returned	0.00035 sec / 0.0000...
47 21:48:53	INSERT INTO Ord...	5 row(s) affected Records: 5 Duplicates: 0...	0.00021 sec
48 21:48:53	SELECT * FROM...	5 row(s) returned	0.00042 sec / 0.0000...
49 21:49:50	INSERT INTO Sal...	4 row(s) affected Records: 4 Duplicates: 0...	0.00023 sec
50 21:49:50	SELECT * FROM...	4 row(s) returned	0.00046 sec / 0.000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Filter objects

> Basic
> Bus_Station
Lab_3
> Tables
Views
Stored Pr...
Functions
> Sample
> sys
> TechCo

Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) Limit to 1000 rows

210 ('Books', 3200.00),
211 ('Furniture', 21000.00);
212
213 • INSERT INTO Top_Students VALUES
214 (1001, 95),
215 (1003, 92),
216 (1005, 98),
217 (1006, 91);
218 • SELECT * FROM Top_Students;
219

100% 28:218

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Student_ID	Top_Marks
1001	95
1003	92
1005	98
1006	91
NULL	NULL

Top_Students 25 Apply

Action Output

Time	Action	Response	Duration / Fetch Time
21:48:53	INSERT INTO ...	5 row(s) affected Records: 5 Duplicates: 0...	0.0021 sec
48	SELECT * FROM...	5 row(s) returned	0.00042 sec / 0.0000...
49	INSERT INTO Sal...	4 row(s) affected Records: 4 Duplicates: 0...	0.0023 sec
50	SELECT * FROM...	4 row(s) returned	0.00046 sec / 0.0000...
51	INSERT INTO Top...	4 row(s) affected Records: 4 Duplicates: 0...	0.0010 sec
52	SELECT * FROM...	4 row(s) returned	0.00031 sec / 0.0000...

Query Completed

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas

SCHEMAS

Q Filter objects

> Basic
> Bus_Station
Lab_3
> Tables
Views
Stored Pr...
Functions
> Sample
> sys
> TechCo

Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1) >

Limit to 1000 rows

(1006, 91);
SELECT * FROM Top_Students;

219
220 • INSERT INTO Customer VALUES
221 (101, 'John', 28, 'Engineer'),
222 (102, 'Alice', 32, 'Doctor'),
223 (103, 'Robert', 40, 'Teacher'),
224 (104, 'Sophia', 25, 'Student'),
225 (105, 'David', 35, 'Businessman');
226 • SELECT * FROM Customer;
227

100% 24:226

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor

Cust_ID	Name	Age	Occupation
101	John	28	Engineer
102	Alice	32	Doctor
103	Robert	40	Teacher
104	Sophia	25	Student
105	David	35	Businessman
HULL	HULL	HULL	HULL

Customer 26 Apply

Action Output

Time	Action	Response	Duration / Fetch Time
21:49:50	INSERT INTO Sam...	4 row(s) affected Records: 4 Duplicates: 0...	0.0023 sec
50	SELECT * FROM...	4 row(s) returned	0.00046 sec / 0.000...
51	INSERT INTO Top...	4 row(s) affected Records: 4 Duplicates: 0...	0.0010 sec
52	SELECT * FROM...	4 row(s) returned	0.00031 sec / 0.0000...
53	INSERT INTO Cu...	5 row(s) affected Records: 5 Duplicates: 0...	0.0020 sec
54	SELECT * FROM...	5 row(s) returned	0.00039 sec / 0.000...

Query Completed

SQL Subquery Questions:

1. Basic **SELECT** Subquery

- Retrieve the names and marks of students whose marks are greater than the average marks.

```
SELECT * FROM Student_Details
```

```
WHERE Stu_Marks > (SELECT AVG(Stu_Marks) FROM Student_Details);
```

The screenshot shows the MySQL Workbench interface. In the left sidebar under 'Schemas', the 'Lab_3' schema is selected. In the main query editor window, a subquery question is displayed with the following code:

```
229 -- Subquery Questions
230 -----
231
232 -- Q1 Retrieve the names and marks of students whose marks are greater
233 -- than the average marks.
234 • SELECT * FROM Student_Details
235 WHERE Stu_Marks > (SELECT AVG(Stu_Marks) FROM Student_Details);
236
```

The code is highlighted with syntax coloring. Below the code, the 'Result Grid' tab is active, showing the results of the query:

Student_RollNo	Stu_Name	Stu_Marks	Stu_City
1003	Bheem	87	Gurgaon
1004	Chetan	95	Noida
1005	Diksha	99	Agra
1006	Raman	90	Ghaziabad
NULL	NULL	NULL	NULL

The 'Action Output' section at the bottom shows the query was executed successfully:

Action	Time	Response	Duration / Fetch Time
1	21:54:15	SELECT * FROM... 4 row(s) returned	0.0028 sec / 0.00001...

A status message 'Query Completed' is visible at the bottom left.

2. IN Operator Subquery

- Fetch the names and addresses of faculties who belong to departments in either 'Noida' or 'Gurgaon.'

```
> SELECT * FROM Department
  WHERE Faculty_ID IN (
    SELECT Faculty_ID FROM Faculty_Details
    WHERE Address = 'Noida' OR Address = 'Gurgaon'
);
```

The screenshot shows the MySQL Workbench interface. The top bar displays the title "MySQL Workbench" and the user information "Divyansh Kumar Singh 590012175". The main window has tabs for "Administration" and "Schemas". The "Schemas" tab is selected, showing a tree view with databases like "Basic", "Bus_Station", "Lab_3" (which is expanded to show "Tables", "Views", "Stored Pr...", "Functions"), "Sample", "sys", and "TechCo". In the central pane, the SQL editor contains the following code:

```
238      -- Q2 Fetch the names and addresses of faculties who belong to
239      -- departments in either 'Noida' or 'Gurgaon.'
240 •   SELECT * FROM Department
241   WHERE Faculty_ID IN (
242     SELECT Faculty_ID FROM Faculty_Details
243     WHERE Address = 'Noida' OR Address = 'Gurgaon'
244   );
245 100% 1:245
```

The code is highlighted with line numbers and comments. The "Result Grid" tab is selected, displaying the results of the query:

	Dept_ID	Faculty_ID	Dept_Name
1	101	BCA	
2	102	B.Tech	
3	105	BBA	
5	107	MCA	
HULL	HULL	HULL	

Below the result grid, the "Action Output" section shows two log entries:

	Time	Action	Response	Duration / Fetch Time
1	21:54:15	SELECT * FROM...	4 row(s) returned	0.0028 sec / 0.00001...
2	22:00:34	SELECT * FROM...	4 row(s) returned	0.0013 sec / 0.00000...

3. INSERT with Subquery

- Insert the details of employees from the "Old_Employee" table into the "New_Employee" table, considering only those with a salary greater than 40000.

```
> INSERT INTO New_Employee
SELECT * FROM Old_Employee
WHERE Emp_Salary > 40000;
```

```
> SELECT * FROM New_Employee;
```

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

- Title Bar:** Divyansh Kumar Singh 590012175
- Schemas Tab:** Shows the current schema is Lab_3, which contains Tables, Views, Stored Procs, and Functions.
- SQL Editor:** Displays the executed SQL code:


```
247 -- Q3 Insert the details of employees from the "Old_Employee" table
248 -- into the "New_Employee" table, considering only those with a salary
249 -- greater than 40000.
250 • INSERT INTO New_Employee
251   SELECT * FROM Old_Employee
252   WHERE Emp_Salary > 40000;
253 • SELECT * FROM New_Employee;
```
- Result Grid:** Shows the data inserted into the New_Employee table:

	Emp_ID	Emp_Name	Emp_Salary	Address
1	1001	Akhil	50000.00	Agra
2	1003	Bheem	45000.00	Gurgaon
3	1004	Cheitan	60000.00	Noida
4	1006	Raman	50000.00	Ghaziabad
5	1008	Sumit	50000.00	Agra
6	1009	Akash	55000.00	Delhi
7	1010	Devansh	65000.00	Gurgaon
	NULL	NULL	NULL	NULL
- Action Output:** Shows the history of actions:

	Time	Action	Response	Duration / Fetch Time
1	21:54:15	SELECT * FROM...	4 row(s) returned	0.0028 sec / 0.00001...
2	22:00:34	SELECT * FROM...	4 row(s) returned	0.0013 sec / 0.00000...
3	22:05:49	INSERT INTO Ne...	4 row(s) affected Records: 4 Duplicates: 0...	0.0025 sec
4	22:05:49	SELECT * FROM...	7 row(s) returned	0.00043 sec / 0.000...

3(b). Another Query using ```INSERT```

```
INSERT INTO New_Employee
SELECT *
FROM Old_Employee
WHERE Emp_ID = ANY (
    SELECT Emp_ID
    FROM Department_Emp
    WHERE Dept_ID IN (406, 407)
);
```

It copies employees from Old_Employee to New_Employee who belong to department 406 or 407.

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

- Title Bar:** MySQL Workbench
- Toolbar:** Standard MySQL Workbench toolbar with icons for file operations, schema navigation, and database connections.
- Schemas Tab:** Shows the current schema is "Lab_3".
- Query Editor:**
 - Text area contains the SQL query provided in the question.
 - Execution status: 100% completed at 3:263.
 - Result grid shows the data copied from the Old_Employee table to the New_Employee table.
- Result Grid:**

Emp_ID	Emp_Name	Emp_Salary	Address
1001	Akhil	50000.00	Agra
1003	Bheem	45000.00	Gurgaon
1004	Chetan	60000.00	Noida
1005	Diksha	30000.00	Agra
1006	Raman	50000.00	Ghaziabad
1008	Sumit	50000.00	Agra
1009	Akash	55000.00	Delhi
1010	Devansh	65000.00	Gurgaon
NULL	NULL	NULL	NULL
- Action Output:**

Time	Action	Response	Duration / Fetch Time
22:21:13	UPDATE Employee_Details S...	2 row(s) affected Rows matched: 2 Change...	0.00052 sec
22:21:17	SELECT * FROM Employee_...	7 row(s) returned	0.00076 sec / 0.0000...
22:24:08	INSERT INTO New_Employe...	1 row(s) affected Records: 1 Duplicates: 0...	0.0018 sec
22:24:08	SELECT * FROM New_Employ...	8 row(s) returned	0.00045 sec / 0.000...
- Status Bar:** Query Completed

4. UPDATE with Subquery:

- Increase the salary of employees by 10% for those whose department grade is 'A.'

> UPDATE Employee_Details

SET Emp_Salary = Emp_Salary + 5000

WHERE Emp_ID IN (SELECT Emp_ID FROM Department_Emp WHERE Dept_Grade = 'A');

> SELECT * FROM Employee_Details;

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

- Title Bar:** MySQL Workbench
- Toolbar:** Standard MySQL Workbench toolbar with icons for Home, Schemas, Tables, Views, Functions, Procedures, Triggers, Events, and Help.
- Schemas Tab:** Shows the current schema is "Lab_3".
- Query Editor:**
 - Text area contains the SQL code for updating employee salaries.
 - Numbered lines 254-261 are visible.
 - Line 257 is highlighted with a blue dot.
 - Line 261 shows the result of the query: "Employee_Details 30".
- Result Grid:** Displays the updated data from the Employee_Details table.

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000.00	404
1002	Balram	25000.00	403
1003	Bheem	50000.00	405
1004	Chetan	65000.00	402
1005	Ram	65000.00	407
1006	Shyam	55500.00	401
1007	Shobhit	60000.00	406
NULL	NULL	NULL	NULL

- Action Output:** Shows the history of database actions with their times, descriptions, responses, and durations.

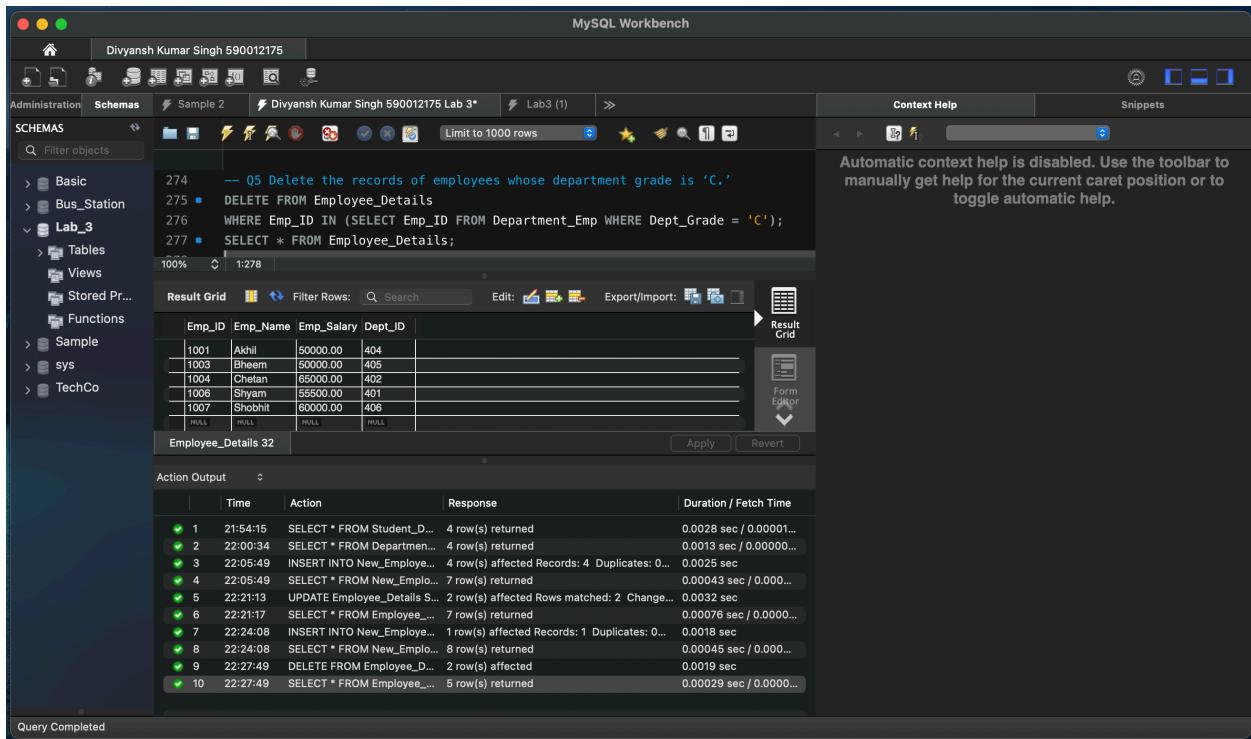
5. DELETE with Subquery:

- Delete the records of employees whose department grade is 'C.'

> `DELETE FROM Employee_Details`

`WHERE Emp_ID IN (SELECT Emp_ID FROM Department_Emp WHERE Dept_Grade = 'C');`

> `SELECT * FROM Employee_Details;`



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

- Session:** Divyansh Kumar Singh 590012175
- Query Editor:**

```

274 -- 05 Delete the records of employees whose department grade is 'C.'
275 • DELETE FROM Employee_Details
276 WHERE Emp_ID IN (SELECT Emp_ID FROM Department_Emp WHERE Dept_Grade = 'C');
277 • SELECT * FROM Employee_Details;
    
```
- Result Grid:** Displays the contents of the Employee_Details table.

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000.00	404
1003	Bheem	50000.00	405
1004	Cheetan	65000.00	402
1006	Shyam	55500.00	401
1007	Shobhit	60000.00	406
NULL	NULL	NULL	NULL

- Action Output:** Shows the history of database actions with their times, descriptions, and durations.

Time	Action	Response	Duration / Fetch Time
21:54:15	SELECT * FROM Student_D...	4 row(s) returned	0.0028 sec / 0.00001...
22:00:34	SELECT * FROM Departmen...	4 row(s) returned	0.0013 sec / 0.00000...
22:05:49	INSERT INTO New_Employe...	4 row(s) affected Records: 4 Duplicates: 0...	0.0025 sec
22:05:49	SELECT * FROM New_Emplo...	7 row(s) returned	0.00043 sec / 0.000...
22:21:13	UPDATE Employee_Details S...	2 row(s) affected Rows matched: 2 Change...	0.0032 sec
22:21:17	SELECT * FROM Employee_...	7 row(s) returned	0.00076 sec / 0.000...
22:24:08	INSERT INTO New_Employe...	1 row(s) affected Records: 1 Duplicates: 0...	0.0018 sec
22:24:08	SELECT * FROM New_Emplo...	8 row(s) returned	0.00045 sec / 0.000...
22:27:49	DELETE FROM Employee_D...	2 row(s) affected	0.0019 sec
22:27:49	SELECT * FROM Employee_...	5 row(s) returned	0.00029 sec / 0.000...

Subquery with Comparison Operator

-- Query 1

```
SELECT *
FROM Employee_Details
WHERE Emp_ID IN (
    SELECT Emp_ID FROM Employee_Details
    WHERE Emp_Salary > 35000
);
```

-- This query selects all employees from Employee_Details whose salary
-- is greater than 35000.

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

- Title Bar:** MySQL Workbench
- Toolbar:** Standard MySQL Workbench toolbar with icons for Home, File, Edit, Tools, Help, etc.
- Schemas Tab:** Shows the current schema is "Lab_3".
- Query Editor:** Displays the SQL code for "Query 1".

```
-- Subquery with Comparison Operator
280  -- Query 1
281 *  SELECT *
282   FROM Employee_Details
283   WHERE Emp_ID IN (
284       SELECT Emp_ID FROM Employee_Details
285       WHERE Emp_Salary > 35000
286   );
287   -- This query selects all employees from Employee_Details whose salary
288   -- is greater than 35000.
```
- Result Grid:** Shows the results of the query, displaying employee details for rows 1001 to 1007.

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000.00	404
1003	Bheem	50000.00	405
1004	Chetan	65000.00	402
1006	Shyam	55500.00	401
1007	Shobhit	60000.00	406
HULL	HULL	HULL	HULL
- Action Output:** Shows the history of actions taken during the session, including insertions, selections, deletions, and other queries.
- Status Bar:** Query Completed

-- Query 2

```
SELECT Emp_Name, Emp_Salary, Dept_ID  
FROM Employee_Details  
WHERE Emp_Salary = (SELECT MAX(Emp_Salary) FROM Employee_Details);  
-- This query shows the employee(s) with the maximum salary in the company.
```

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

- Title Bar:** MySQL Workbench
- Toolbar:** Standard MySQL Workbench toolbar with various icons for database management.
- Schemas Tab:** Shows the current schema is "Lab_3".
- Query Editor:** Contains the SQL code for Query 2, including the explanatory comment at the end.
- Result Grid:** Displays the results of the query, showing one row for Chetan with Emp_Salary 65000.00 and Dept_ID 402.
- Action Output:** Shows the history of database operations, including insertions, selections, updates, and deletions, along with their times and durations.
- Status Bar:** Shows "Query Completed" at the bottom.

6. Subquery in WHERE Clause - NOT IN

- Retrieve the names of students who do not belong to the city 'Los Angeles' based on data from the "Student2" table.

```
SELECT Name, City FROM Student
```

```
WHERE City NOT IN (SELECT City FROM Student2 WHERE City = 'Los Angeles');
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, with 'Lab_3' selected. The main pane shows a SQL editor with the following code:

```
296
297      -- Q6 Retrieve the names of students who do not belong to the city 'Los
298      -- Angeles' based on data from the "Student2" table.
299 *  SELECT Name, City FROM Student
300 WHERE City NOT IN (SELECT City FROM Student2 WHERE City = 'Los Angeles');
301
```

The results grid shows the following data:

Name	City
Peter	Texas
Suzi	California
Joseph	Alaska
Brayan	New York

The bottom pane shows the 'Action Output' history:

Action	Time	Response	Duration / Fetch Time
SELECT * FROM New_Employees	22:05:49	7 row(s) returned	0.00043 sec / 0.000...
UPDATE Employee_Details S...	22:21:13	2 row(s) affected Rows matched: 2 Change...	0.0032 sec
SELECT * FROM Employee_...	22:21:17	7 row(s) returned	0.00076 sec / 0.000...
INSERT INTO New_Employ...	22:24:08	1 row(s) affected Records: 1 Duplicates: 0...	0.0018 sec
SELECT * FROM New_Emplo...	22:24:08	8 row(s) returned	0.00045 sec / 0.000...
DELETE FROM Employee_D...	22:27:49	2 row(s) affected	0.0019 sec
SELECT * FROM Employee_...	22:27:49	5 row(s) returned	0.00029 sec / 0.0000...
SELECT * FROM Employee_...	22:34:19	5 row(s) returned	0.0011 sec / 0.00001...
SELECT emp_name, city, inc...	22:36:26	Error Code: 1146. Table 'lab_3.employees' d...	0.0021 sec
SELECT Emp_Name, Emp_S...	22:37:22	1 row(s) returned	0.00087 sec / 0.0000...
SELECT Name, City FROM S...	22:38:25	4 row(s) returned	0.00093 sec / 0.000...

The status bar at the bottom indicates 'Query Completed'.

7. Subquery in **FROM** Clause:

- Use a subquery in the FROM clause to create a derived table showing the maximum, minimum, and average number of items for each order.

```
SELECT MAX(items), MIN(items), FLOOR(AVG(items))
```

```
FROM (
    SELECT COUNT(Order_ID) AS items
    FROM Orders
    GROUP BY Order_Date
) AS OrderSummary;
```

MySQL Workbench

Divyansh Kumar Singh 590012175

Administration Schemas Sample 2 Divyansh Kumar Singh 590012175 Lab 3* Lab3 (1)

SCHEMAS Filter objects

Basic Bus_Station Lab_3 Tables Views Stored Pr... Functions Sample sys TechCo

-- Q7 Use a subquery in the FROM clause to create a derived table showing
-- the maximum, minimum, and average number of items for each order.
303 • SELECT MAX(items), MIN(items), FLOOR(AVG(items))
304 • FROM (
305 SELECT COUNT(Order_ID) AS items
306 FROM Orders
307 GROUP BY Order_Date
308)
309 AS OrderSummary;

Result Grid Filter Rows: Search Export: Result Grid

	MAX(items)	MIN(items)	FLOOR(AVG(items))
1	1	1	
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			

Result 36 Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
8	22:24:08	SELECT * FROM New_Emplo...	8 row(s) returned	0.00045 sec / 0.000...
9	22:27:49	DELETE FROM Employee_D...	2 row(s) affected	0.0019 sec
10	22:27:49	SELECT * FROM Employee_...	5 row(s) returned	0.00029 sec / 0.0000...
11	22:34:19	SELECT * FROM Employee_...	5 row(s) returned	0.0011 sec / 0.00001...
12	22:36:26	SELECT emp_name, city, inc...	Error Code: 1146. Table 'lab_3.employees' d...	0.0021 sec
13	22:37:22	SELECT Emp_Name, Emp_S...	1 row(s) returned	0.00087 sec / 0.0000...
14	22:38:25	SELECT Name, City FROM S...	4 row(s) returned	0.00093 sec / 0.000...
15	22:40:54	SELECT MAX(items), MIN(it...	Error Code: 1055. Expression #1 of SELECT...	0.0011 sec
16	22:41:27	SELECT MAX(items), MIN(it...	1 row(s) returned	0.00098 sec / 0.000...

Query Completed

8. Correlated Subquery

- Fetch the names, cities, and incomes of employees whose income is higher than the average income of employees in their respective cities.

```
SELECT Emp_Name, Dept_ID, Emp_Salary
FROM Employee_Details e
WHERE Emp_Salary > (
    SELECT AVG(Emp_Salary)
    FROM Employee_Details
    WHERE Dept_ID = e.Dept_ID
);
```

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

- Title Bar:** MySQL Workbench, Divyansh Kumar Singh 590012175
- Schemas Tab:** Shows the current schema is **Lab_3**.
- Query Editor:** Displays the SQL query:


```
-- Q8 Fetch the names, cities, and incomes of employees whose income is
-- higher than the average income of employees in their respective cities.
313 •   SELECT Emp_Name, Dept_ID, Emp_Salary
314     FROM Employee_Details e
315   WHERE Emp_Salary > (
316       SELECT AVG(Emp_Salary)
317       FROM Employee_Details
318       WHERE Dept_ID = e.Dept_ID
319   );
320 
```
- Result Grid:** A table titled "Employee_Details 38" with columns Emp_Name, Dept_ID, and Emp_Salary. The grid is currently empty.
- Action Output:** A table showing the history of actions:

	Time	Action	Response	Duration / Fetch Time
✓ 10	22:27:49	SELECT * FROM Employee_...	5 row(s) returned	0.00029 sec / 0.0000...
✓ 11	22:34:19	SELECT * FROM Employee_...	5 row(s) returned	0.0011 sec / 0.00001...
✗ 12	22:36:26	SELECT emp_name, city, inc...	Error Code: 1146. Table 'lab_3.employees' d...	0.0021 sec
✓ 13	22:37:22	SELECT Emp_Name, Emp_S...	1 row(s) returned	0.00087 sec / 0.000...
✓ 14	22:38:25	SELECT Name, City FROM S...	4 row(s) returned	0.00093 sec / 0.000...
✗ 15	22:40:54	SELECT MAX(items), MIN(it...	Error Code: 1055. Expression #1 of SELECT...	0.0011 sec
✓ 16	22:41:27	SELECT MAX(items), MIN(it...	1 row(s) returned	0.00098 sec / 0.000...
✓ 17	22:45:47	SELECT Emp_Name, Dept_I...	0 row(s) returned	0.00097 sec / 0.0000...
✓ 18	22:46:07	SELECT Emp_Name, Dept_I...	0 row(s) returned	0.00081 sec / 0.0000...
- Status Bar:** Query Completed

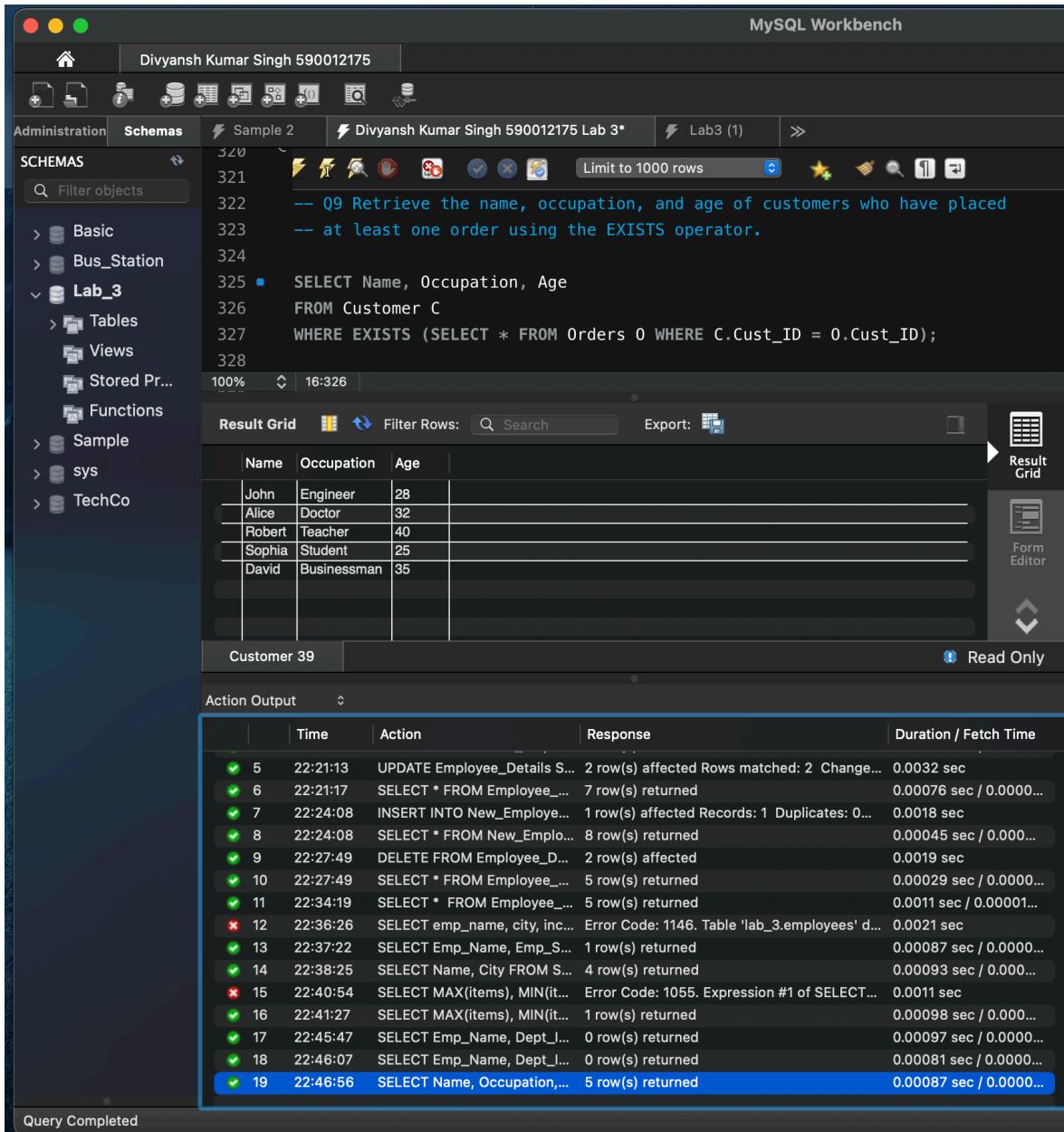
9. EXISTS Operator

- Retrieve the name, occupation, and age of customers who have placed at least one order using the EXISTS operator.

`SELECT Name, Occupation, Age`

`FROM Customer C`

`WHERE EXISTS (SELECT * FROM Orders O WHERE C.Cust_ID = O.Cust_ID);`



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

- Title Bar:** MySQL Workbench, Divyansh Kumar Singh 590012175
- Toolbar:** Standard MySQL Workbench tools like Home, File, Edit, Tools, Help, etc.
- Schemas Tab:** Shows the current schema is `Lab_3`.
- Query Editor:**
 - Text area contains the SQL query:


```
-- Q9 Retrieve the name, occupation, and age of customers who have placed
-- at least one order using the EXISTS operator.

325 •  SELECT Name, Occupation, Age
326   FROM Customer C
327 WHERE EXISTS (SELECT * FROM Orders O WHERE C.Cust_ID = O.Cust_ID);
328
```
 - Status bar shows 100% completion and the time 16:326.
- Result Grid:** Displays the query results in a table format:

Name	Occupation	Age
John	Engineer	28
Alice	Doctor	32
Robert	Teacher	40
Sophia	Student	25
David	Businessman	35
- Action Output:** Shows a history of database actions with their times, descriptions, responses, and durations. Action 19 is highlighted in blue:

	Time	Action	Response	Duration / Fetch Time
✓ 5	22:21:13	UPDATE Employee_Details S...	2 row(s) affected Rows matched: 2 Change...	0.0032 sec
✓ 6	22:21:17	SELECT * FROM Employee_...	7 row(s) returned	0.00076 sec / 0.0000...
✓ 7	22:24:08	INSERT INTO New_Employee...	1 row(s) affected Records: 1 Duplicates: 0...	0.0018 sec
✓ 8	22:24:08	SELECT * FROM New_Emplo...	8 row(s) returned	0.00045 sec / 0.000...
✓ 9	22:27:49	DELETE FROM Employee_...	2 row(s) affected	0.0019 sec
✓ 10	22:27:49	SELECT * FROM Employee_...	5 row(s) returned	0.00029 sec / 0.0000...
✓ 11	22:34:19	SELECT * FROM Employee_...	5 row(s) returned	0.0011 sec / 0.00001...
✗ 12	22:36:26	SELECT emp_name, city, inc...	Error Code: 1146. Table 'lab_3.employees' d...	0.0021 sec
✓ 13	22:37:22	SELECT Emp_Name, Emp_S...	1 row(s) returned	0.00087 sec / 0.0000...
✓ 14	22:38:25	SELECT Name, City FROM S...	4 row(s) returned	0.00093 sec / 0.000...
✗ 15	22:40:54	SELECT MAX(items), MIN(it...	Error Code: 1055. Expression #1 of SELECT...	0.0011 sec
✓ 16	22:41:27	SELECT MAX(items), MIN(it...	1 row(s) returned	0.00098 sec / 0.000...
✓ 17	22:45:47	SELECT Emp_Name, Dept_I...	0 row(s) returned	0.00097 sec / 0.0000...
✓ 18	22:46:07	SELECT Emp_Name, Dept_I...	0 row(s) returned	0.00081 sec / 0.0000...
✓ 19	22:46:56	SELECT Name, Occupation,...	5 row(s) returned	0.00087 sec / 0.0000...
- Status Bar:** Query Completed

10. ROW Subquery

- Retrieve all columns for customers whose (cust_id, occupation) matches any row of (order_id, order_date) from the "Orders" table.

```
SELECT * FROM Customer C
WHERE ROW(Cust_ID, Occupation) =
    (SELECT Order_ID, Order_Date FROM Orders O WHERE C.Cust_ID =
O.Cust_ID
);
```

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

- Title Bar:** MySQL Workbench
- Toolbar:** Standard MySQL Workbench toolbar with various icons for database management.
- Schemas Tab:** Shows the current schema is 'Lab_3'.
- Query Editor:**
 - Text area: The SQL query provided in the question.
 - Execution status: The query was executed successfully, indicated by a green checkmark icon.
 - Output pane: Shows the results of the query, which consists of 20 rows of data.
- Results Table:**

	Time	Action	Response	Duration / Fetch Time
✓ 6	22:21:17	SELECT * FROM Employee_...	7 row(s) returned	0.00076 sec / 0.0000...
✓ 7	22:24:08	INSERT INTO New_Employee...	1 row(s) affected Records: 1 Duplicates: 0...	0.0018 sec
✓ 8	22:24:08	SELECT * FROM New_Emplo...	8 row(s) returned	0.00045 sec / 0.000...
✓ 9	22:27:49	DELETE FROM Employee_D...	2 row(s) affected	0.0019 sec
✓ 10	22:27:49	SELECT * FROM Employee_...	5 row(s) returned	0.00029 sec / 0.0000...
✓ 11	22:34:19	SELECT * FROM Employee_...	5 row(s) returned	0.0011 sec / 0.00001...
✗ 12	22:36:26	SELECT emp_name, city, inc...	Error Code: 1146. Table 'lab_3.employees' d...	0.0021 sec
✓ 13	22:37:22	SELECT Emp_Name, Emp_S...	1 row(s) returned	0.00087 sec / 0.0000...
✓ 14	22:38:25	SELECT Name, City FROM S...	4 row(s) returned	0.00093 sec / 0.000...
✗ 15	22:40:54	SELECT MAX(items), MIN(it...	Error Code: 1055. Expression #1 of SELECT...	0.0011 sec
✓ 16	22:41:27	SELECT MAX(items), MIN(it...	1 row(s) returned	0.00098 sec / 0.000...
✓ 17	22:45:47	SELECT Emp_Name, Dept_I...	0 row(s) returned	0.00097 sec / 0.0000...
✓ 18	22:46:07	SELECT Emp_Name, Dept_I...	0 row(s) returned	0.00081 sec / 0.0000...
✓ 19	22:46:56	SELECT Name, Occupation,...	5 row(s) returned	0.00087 sec / 0.0000...
✓ 20	22:48:32	SELECT * FROM Customer C...	0 row(s) returned	0.0013 sec / 0.00001...
- Status Bar:** Query Completed

11. Subquery with ALL Operator

- Find customers whose cust_id is greater than all cust_ids in the "Orders" table.

```
SELECT Cust_ID, Name FROM Customer  
WHERE Cust_ID > ALL (SELECT Cust_ID FROM Orders);
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, with 'Lab_3' selected. The main pane contains the following SQL code:

```
335  
336 -- Q11 Find customers whose cust_id is greater than all cust_ids in the  
337 -- "Orders" table.  
338  
339 • SELECT Cust_ID, Name FROM Customer  
340 WHERE Cust_ID > ALL (SELECT Cust_ID FROM Orders);  
341
```

The 'Result Grid' tab is active, showing the output of the query:

Cust_ID	Name
NULL	NULL

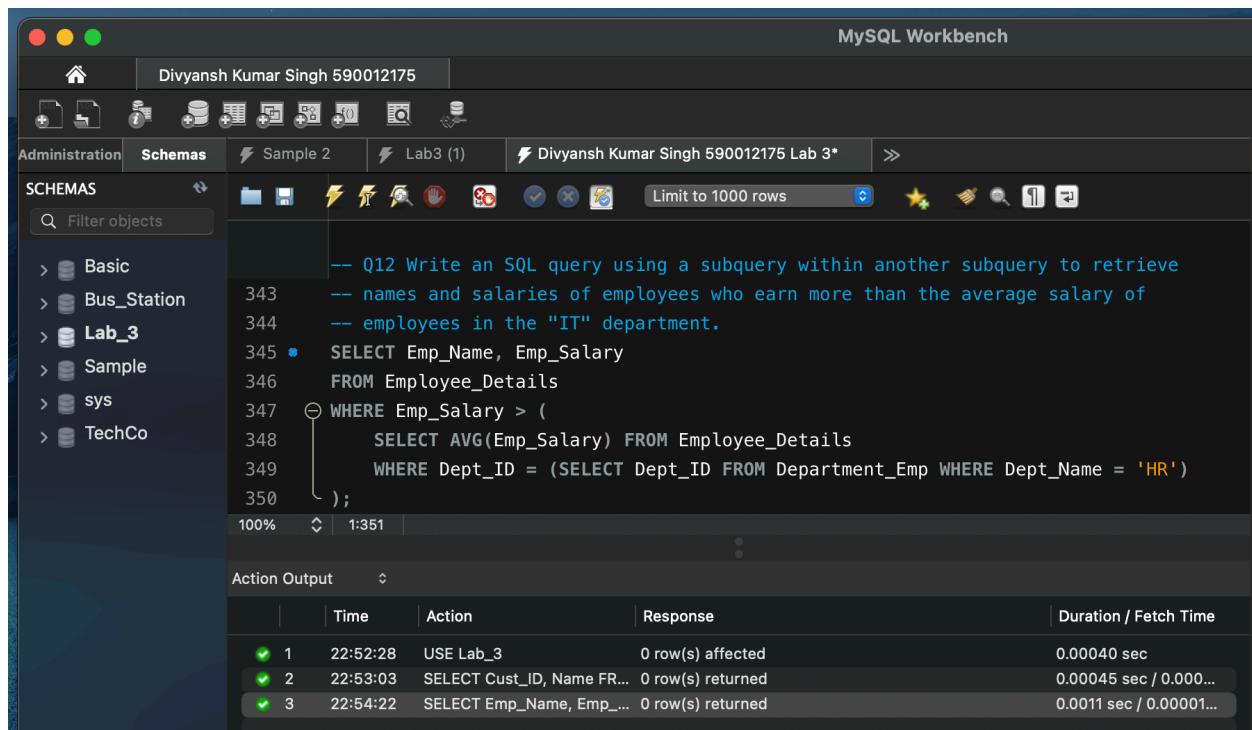
Below the result grid, the 'Action Output' section shows the following log entries:

Time	Action	Response	Duration / Fetch Time
22:52:28	USE Lab_3	0 row(s) affected	0.00040 sec
22:53:03	SELECT Cust_ID,...	0 row(s) returned	0.00045 sec / 0.000...

12. Nested Subqueries

- Write an SQL query using a subquery within another subquery to retrieve names and salaries of employees who earn more than the average salary of employees in the "IT" department.

```
SELECT Emp_Name, Emp_Salary
FROM Employee_Details
WHERE Emp_Salary > (
    SELECT AVG(Emp_Salary) FROM Employee_Details
    WHERE Dept_ID = (SELECT Dept_ID FROM Department_Emp
    WHERE Dept_Name = 'HR')
);
```



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

- Title Bar:** MySQL Workbench
- Session Information:** Divyansh Kumar Singh 590012175
- Schemas Tab:** Shows available databases: Basic, Bus_Station, Lab_3, Sample, sys, TechCo. The Lab_3 database is selected.
- SQL Editor:** Contains the SQL query provided in the question. The line numbers 343, 344, and 345 are highlighted, indicating the part of the query being executed.
- Action Output:** Displays the execution results in a table format.

	Time	Action	Response	Duration / Fetch Time
1	22:52:28	USE Lab_3	0 row(s) affected	0.00040 sec
2	22:53:03	SELECT Cust_ID, Name FR...	0 row(s) returned	0.00045 sec / 0.000...
3	22:54:22	SELECT Emp_Name, Emp...	0 row(s) returned	0.0011 sec / 0.00001...

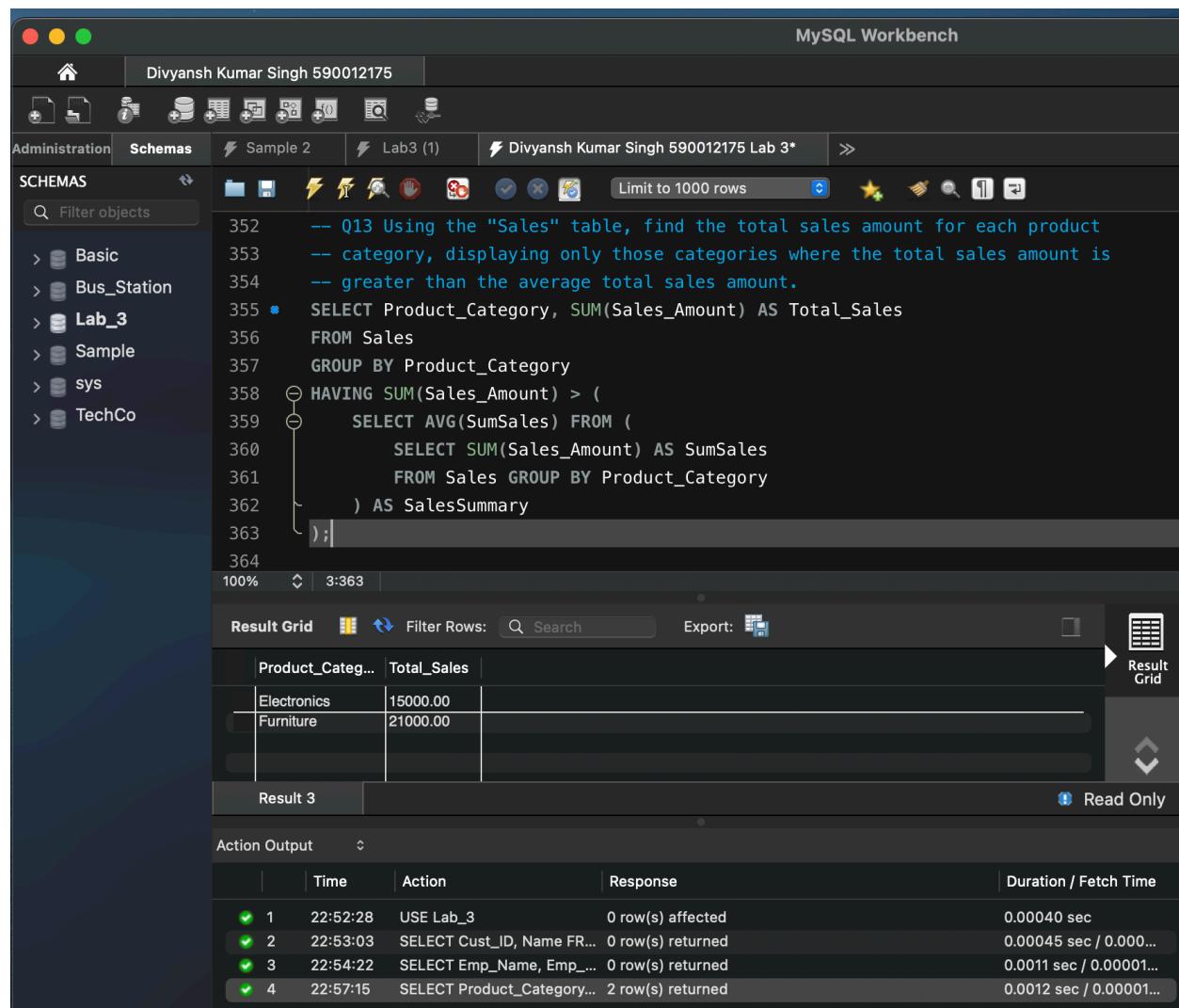
13. Subquery in HAVING Clause

- Using the "Sales" table, find the total sales amount for each product category, displaying only those categories where the total sales amount is greater than the average total sales amount.

```

SELECT Product_Category, SUM(Sales_Amount) AS Total_Sales
FROM Sales
GROUP BY Product_Category
HAVING SUM(Sales_Amount) > (
    SELECT AVG(SumSales) FROM (
        SELECT SUM(Sales_Amount) AS SumSales
        FROM Sales GROUP BY Product_Category
    ) AS SalesSummary
);

```



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

- Title Bar:** MySQL Workbench
- Toolbar:** Standard MySQL Workbench toolbar with various icons for database management.
- Schemas Tab:** Shows the current schema is "Lab_3".
- Query Editor:**
 - Text area contains the SQL query provided above.
 - Line numbers 352 to 364 are visible on the left.
 - Execution status: Line 355 is highlighted with a blue dot, indicating it's the current step.
 - Result grid below the editor shows the output:
- Result Grid:**

Product_Categ...	Total_Sales
Electronics	15000.00
Furniture	21000.00
- Action Output:** Shows the history of actions and their durations.

14. Subquery with GROUP BY

- Retrieve department names and the count of employees in each department, filtering out departments where the count is less than 5.

```
SELECT D.Dept_Name, COUNT(E.Emp_ID) AS EmployeeCount
FROM Employee_Details E
JOIN Department_Emp D ON E.Dept_ID = D.Dept_ID
GROUP BY D.Dept_Name
HAVING COUNT(E.Emp_ID) >= 5;
```

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

- Session Title:** Divyansh Kumar Singh 590012175
- Schemas:** Lab_3 (selected)
- Query Editor:**
 - Text area: The provided SQL query for Q14.
 - Line numbers: 363 to 373.
 - Status bar: 100% completion, 29:372 time.
- Result Grid:**

Dept_Name	EmployeeCount

Result 4 | Read Only
- Action Output:**

	Time	Action	Response	Duration / Fetch Time
1	22:52:28	USE Lab_3	0 row(s) affected	0.00040 sec
2	22:53:03	SELECT Cust_ID, Name FR...	0 row(s) returned	0.00045 sec / 0.000...
3	22:54:22	SELECT Emp_Name, Emp_...	0 row(s) returned	0.0011 sec / 0.00001...
4	22:57:15	SELECT Product_Category...	2 row(s) returned	0.0012 sec / 0.00001...
5	22:58:38	SELECT D.Dept_Name, C...	0 row(s) returned	0.0011 sec / 0.00000...

15. Multiple Subqueries in a Single Query

- Write an SQL query involving multiple subqueries to retrieve information about employees based on various conditions.

```
SELECT Emp_Name, Emp_Salary, Dept_ID
FROM Employee_Details
WHERE Emp_Salary > (SELECT AVG(Emp_Salary) FROM Employee_Details)
AND Dept_ID IN (SELECT Dept_ID FROM Department_Emp WHERE
Dept_Grade = 'A');
```

The screenshot shows the MySQL Workbench interface. The top bar displays the title "MySQL Workbench" and the user "Divyansh Kumar Singh 590012175". The left sidebar shows the "Schemas" tree with "Lab_3" selected. The main pane contains the following text:

```
-- Q15 Write an SQL query involving multiple subqueries to retrieve information
-- about employees based on various conditions.

• 377   SELECT Emp_Name, Emp_Salary, Dept_ID
378     FROM Employee_Details
379 WHERE Emp_Salary > (SELECT AVG(Emp_Salary) FROM Employee_Details)
380       AND Dept_ID IN (SELECT Dept_ID FROM Department_Emp WHERE Dept_Grade = 'A');

100% | 1:381 |
```

The "Result Grid" section below shows the query results:

Emp_Name	Emp_Salary	Dept_ID
Chetan	65000.00	402

The "Action Output" section at the bottom lists the execution steps:

Time	Action	Response	Duration / Fetch Time
22:52:28	USE Lab_3	0 row(s) affected	0.00040 sec
22:53:03	SELECT Cust_ID, Name FR...	0 row(s) returned	0.00045 sec / 0.000...
22:54:22	SELECT Emp_Name, Emp_...	0 row(s) returned	0.0011 sec / 0.00001...
22:57:15	SELECT Product_Categori...	2 row(s) returned	0.0012 sec / 0.00001...
22:58:38	SELECT D.Dept_Name, C...	0 row(s) returned	0.0011 sec / 0.00000...
23:00:43	SELECT Emp_Name, Emp_...	1 row(s) returned	0.0015 sec / 0.00001...

16. Subquery with BETWEEN Operators

- Find the names and marks of students who scored between 80 and 90.

```
SELECT Stu_Name, Stu_Marks  
FROM Student_Details  
WHERE Stu_Marks BETWEEN 80 AND 90;
```

The screenshot shows the MySQL Workbench interface. The top bar displays the title "MySQL Workbench" and the user information "Divyansh Kumar Singh 590012175". The main area shows a list of queries in the SQL editor. The third query is selected and contains the code provided above. The result grid below shows three rows of data: Akhil (85), Bheem (87), and Raman (90). The bottom section shows the "Action Output" table with seven log entries, including the execution of the current query.

Time	Action	Response	Duration / Fetch Time
22:52:28	USE Lab_3	0 row(s) affected	0.00040 sec
22:53:03	SELECT Cust_ID, Name FR...	0 row(s) returned	0.00045 sec / 0.000...
22:54:22	SELECT Emp_Name, Emp_...	0 row(s) returned	0.0011 sec / 0.00001...
22:57:15	SELECT Product_Categori...	2 row(s) returned	0.0012 sec / 0.00001...
22:58:38	SELECT D.Dept_Name, C...	0 row(s) returned	0.0011 sec / 0.0000...
23:00:43	SELECT Emp_Name, Emp_...	1 row(s) returned	0.0015 sec / 0.00001...
23:01:33	SELECT Stu_Name, Stu_M...	3 row(s) returned	0.00092 sec / 0.000...

17. Subquery in INSERT Statement

- Insert the details of employees from the "Old_Employee" table into the "New_Employee" table, considering only those with a salary greater than the average salary of all employees.

```
> INSERT INTO New_Employee (Emp_ID, Emp_Name, Emp_Salary, Address)
```

```
SELECT o.Emp_ID, o.Emp_Name, o.Emp_Salary, o.Address
FROM Old_Employee o
WHERE Emp_Salary > (SELECT AVG(Emp_Salary) FROM Old_Employee)
```

```
ON DUPLICATE KEY UPDATE
```

```
    Emp_Name = o.Emp_Name,
    Emp_Salary = o.Emp_Salary,
    Address = o.Address;
```

```
> SELECT * FROM New_Employee;
```

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

- Title Bar:** Divyansh Kumar Singh 590012175
- Schemas Tab:** Shows the current schema is Lab_3.
- SQL Editor:**
 - Text area contains the SQL query for inserting data into the New_Employee table based on a subquery for the average salary.
 - Line 399 shows the final part of the query: "SELECT * FROM New_Employee;"
 - Execution status: 100% completed at 28:399.
- Result Grid:**

Emp_ID	Emp_Name	Emp_Salary	Address
1001	Akhil	50000.00	Agra
1003	Bheem	45000.00	Gurgaon
1004	Chetan	60000.00	Noida
1005	Diksha	30000.00	Agra
1006	Raman	50000.00	Ghaziabad
1008	Sumit	50000.00	Agra
1009	Akash	55000.00	Delhi
1010	Devansh	65000.00	Gurgaon
HULL	HULL	HULL	HULE
- Action Output:**

Action	Time	Response	Duration / Fetch Time
5	22:58:38	SELECT D.Dept_Name, C...	0.0011 sec / 0.0000...
6	23:00:43	SELECT Emp_Name, Emp...	0.0015 sec / 0.00001...
7	23:01:33	SELECT Stu_Name, Stu_M...	0.00092 sec / 0.0000...
8	23:02:20	INSERT INTO New_Employ...	Error Code: 1062. Duplicate entry '1001' for key 'new_...'
9	23:03:56	INSERT INTO New_Employ...	0 row(s) affected, 3 warning(s): 1287 "VALUES functi...
10	23:04:31	INSERT INTO New_Employ...	0 row(s) affected Records: 4 Duplicates: 0 Warnings: 0
11	23:05:06	SELECT * FROM New_Emp...	0.00071 sec / 0.0000...

18. Subquery with ANY Operator

- Retrieve the names of students who scored more marks than ANY student from the "Top_Students" table.

```
SELECT Stu_Name
FROM Student_Details
WHERE Stu_Marks > ANY (SELECT Top_Marks FROM Top_Students);
```

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

```
-- Q18 Retrieve the names of students who scored more marks than ANY student
-- from the "Top_Students" table.
SELECT Stu_Name
FROM Student_Details
WHERE Stu_Marks > ANY (SELECT Top_Marks FROM Top_Students);
```

The results grid displays two rows of data:

Stu_Name
Chetan
Diksha

The Action Output pane shows the history of database operations:

Action	Time	Response	Duration / Fetch Time
USE Lab_3	22:52:28	0 row(s) affected	0.00040 sec
SELECT Cust_ID, Name FR...	22:53:03	0 row(s) returned	0.00045 sec / 0.000...
SELECT Emp_Name, Emp_...	22:54:22	0 row(s) returned	0.0011 sec / 0.00001...
SELECT Product_Categori...	22:57:15	2 row(s) returned	0.0012 sec / 0.00001...
SELECT D.Dept_Name, C...	22:58:38	0 row(s) returned	0.0011 sec / 0.00000...
SELECT Emp_Name, Emp_...	23:00:43	1 row(s) returned	0.0015 sec / 0.00001...
SELECT Stu_Name, Stu_M...	23:01:33	3 row(s) returned	0.00092 sec / 0.0000...
INSERT INTO New_Employ...	23:02:20	Error Code: 1062. Duplicate entry '1001' for key 'new_...'	0.0013 sec
INSERT INTO New_Employ...	23:03:56	0 row(s) affected, 3 warning(s): 1287 'VALUES functi...	0.00098 sec
INSERT INTO New_Employ...	23:04:31	0 row(s) affected Records: 4 Duplicates: 0 Warnings...	0.0010 sec
SELECT * FROM New_Emp...	23:05:06	8 row(s) returned	0.00071 sec / 0.0000...
SELECT Stu_Name FROM...	23:06:51	2 row(s) returned	0.00098 sec / 0.000...

19. Subquery in UPDATE Statement

- Update the salary of employees by 5% for those whose department is 'HR' and the salary is less than the average salary of HR department employees.

```
UPDATE Employee_Details e
```

```
JOIN (
```

```
    SELECT AVG(Emp_Salary) AS avg_salary
    FROM Employee_Details
```

```
        WHERE Dept_ID = (SELECT Dept_ID FROM Department_Emp
        WHERE Dept_Name = 'HR')
```

```
) t
```

```
SET e.Emp_Salary = e.Emp_Salary * 1.05
```

```
WHERE e.Dept_ID = (SELECT Dept_ID FROM Department_Emp WHERE
Dept_Name = 'HR')
```

```
AND e.Emp_Salary < t.avg_salary;
```

```
SELECT * FROM Employee_Details;
```

MySQL Workbench

Administration Schemas Sample 2 Lab3 (1) Divyansh Kumar Singh 590012175 Lab 3* >

SCHEMAS Basic Bus_Station Lab_3 Sample sys TechCo

408 -- Q19 Update the salary of employees by 5% for those whose department is 'HR'
409 -- and the salary is less than the average salary of HR department employees.
410
411 • UPDATE Employee_Details e
412 ⊕ JOIN (

SELECT AVG(Emp_Salary) AS avg_salary
FROM Employee_Details
WHERE Dept_ID = (SELECT Dept_ID FROM Department_Emp WHERE Dept_Name = 'HR'))
416) t
417 SET e.Emp_Salary = e.Emp_Salary * 1.05
418 WHERE e.Dept_ID = (SELECT Dept_ID FROM Department_Emp WHERE Dept_Name = 'HR')
419 AND e.Emp_Salary < t.avg_salary;
420
421 • | SELECT * FROM Employee_Details;

100% 32:421

Result Grid | Filter Rows: Search Edit: Export/Import: Result Grid

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000.00	404
1003	Bheem	50000.00	405
1004	Chetan	65000.00	402
1006	Shyam	55500.00	401
1007	Shobhit	60000.00	406
HULL	HULL	HULL	HULL

Employee_Details 9 Apply Revert

Action Output

Time	Action	Response	Duration / Fetch Time
7	23:01:33	SELECT Stu_Name, Stu_M...	3 row(s) returned 0.00092 sec / 0.0000...
8	23:02:20	INSERT INTO New_Employ...	Error Code: 1062. Duplicate entry '1001' for key 'new_...' 0.0013 sec
9	23:03:56	INSERT INTO New_Employ...	0 row(s) affected, 3 warning(s): 1287 'VALUES functi... 0.00010 sec
10	23:04:31	INSERT INTO New_Employ...	0 row(s) affected Records: 4 Duplicates: 0 Warnings: 0 0.00071 sec / 0.0000...
11	23:05:06	SELECT * FROM New_Emp...	8 row(s) returned 0.000098 sec / 0.0000...
12	23:06:51	SELECT Stu_Name FROM...	2 row(s) returned 0.000098 sec / 0.0000...
13	23:08:17	UPDATE Employee_Details...	Error Code: 1093. You can't specify target table 'Empl...' 0.00010 sec
14	23:09:35	UPDATE Employee_Details...	0 row(s) affected Rows matched: 0 Changed: 0 War... 0.00012 sec
15	23:09:35	SELECT * FROM Employee...	5 row(s) returned 0.000031 sec / 0.0000...

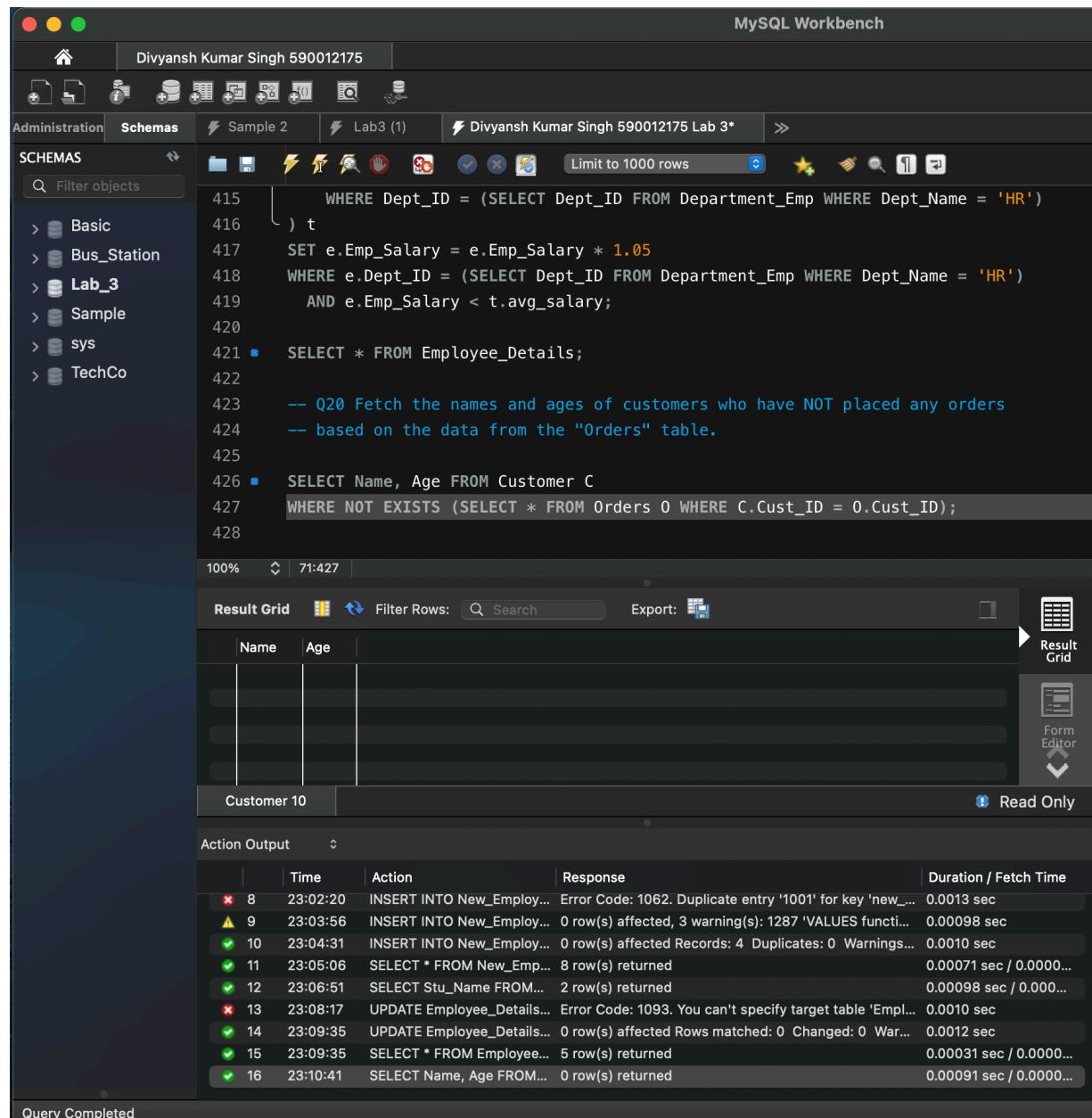
Query Completed

20. Subquery with NOT EXISTS

- Fetch the names and ages of customers who have NOT placed any orders based on the data from the "Orders" table.

`SELECT Name, Age FROM Customer C`

`WHERE NOT EXISTS (SELECT * FROM Orders O WHERE C.Cust_ID = O.Cust_ID);`



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

```

415     WHERE Dept_ID = (SELECT Dept_ID FROM Department_Emp WHERE Dept_Name = 'HR')
416   ) t
417   SET e.Emp_Salary = e.Emp_Salary * 1.05
418   WHERE e.Dept_ID = (SELECT Dept_ID FROM Department_Emp WHERE Dept_Name = 'HR')
419       AND e.Emp_Salary < t.avg_salary;
420
421 •   SELECT * FROM Employee_Details;
422
423 -- Q20 Fetch the names and ages of customers who have NOT placed any orders
424 -- based on the data from the "Orders" table.
425
426 •   SELECT Name, Age FROM Customer C
427 WHERE NOT EXISTS (SELECT * FROM Orders O WHERE C.Cust_ID = O.Cust_ID);
428

```

The results grid below the editor shows a single row with columns 'Name' and 'Age'. The status bar at the bottom indicates 'Query Completed'.

CONCLUSION

In this lab, we successfully implemented different types of SQL subqueries using **SELECT**, **INSERT**, **UPDATE**, and **DELETE**.

We explored operators like **IN**, **ANY**, **ALL**, **EXISTS**, and **NOT EXISTS** with practical table data.

The exercise demonstrated the power of subqueries in filtering, aggregating, and updating data efficiently.

We also learned how correlated and nested subqueries provide deeper insights in complex queries.

Overall, this lab enhanced our understanding of SQL subqueries and their applications in real-world database operations.