

School of Engineering & Technology
Department of Computer Science & Technology



Data Base Management System Lab
(CSE249)

Lab File
(2024-2025)

for

B. Tech. (CSE)
4th Semester

Submitted To:

Dr. Abhishek Singh Verma
Associate Professor,
Department of Computer Science &
Engineering
School of Engineering & Technology
Sharda University

Submitted By:

Masood Aslam
B. Tech. CSE [4th Semester]
2301010501
Group 1

EXPERIMENT-1

OBJECTIVE : Program to Create Table.

SOFTWARE USED : MySQL Workbench 8.0 CE

THEORY

CREATE : It is a DDL command. It is used to create table in SQL.

SYNTAX

CREATE TABLE Table_name

(

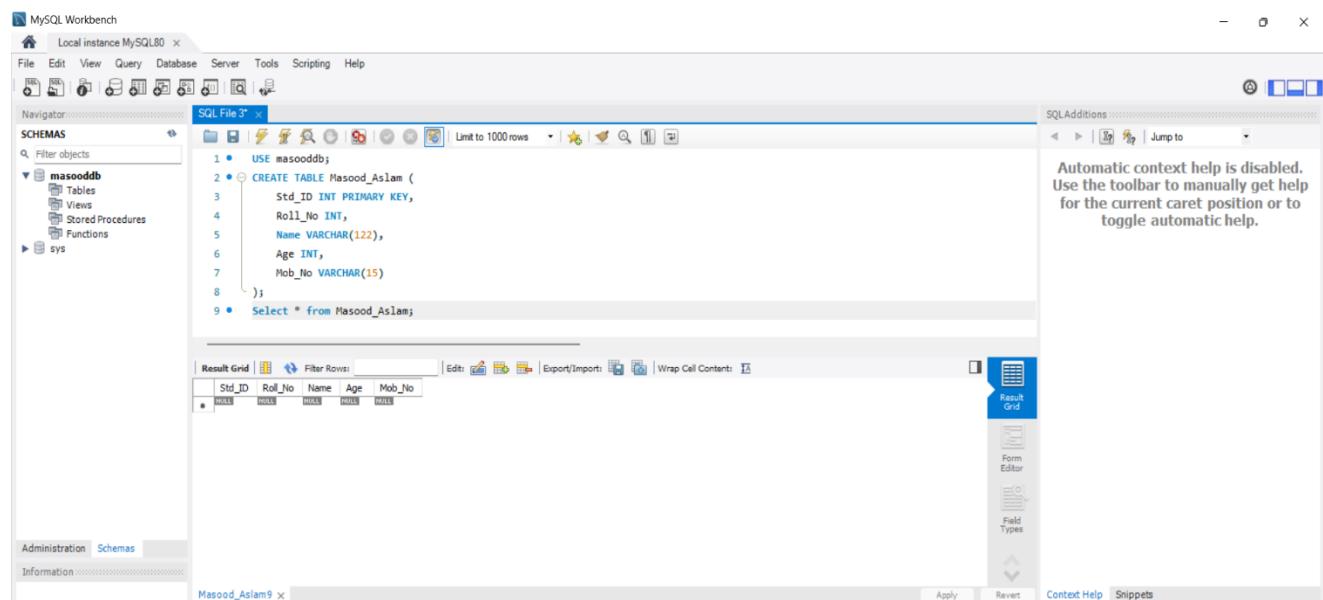
Attribute_name1 Data_type(Size),

Attribute_name2 Data_type(Size),

.....

Attribute_name3 Data_type(Size)

);



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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (masooldb selected), Tables, Views, Stored Procedures, Functions, sys.
- SQL Editor:**

```

1 USE masooldb;
2 CREATE TABLE Masood_Aslam (
3     Std_ID INT PRIMARY KEY,
4     Roll_No INT,
5     Name VARCHAR(122),
6     Age INT,
7     Mob_No VARCHAR(15)
8 );
9 Select * from Masood_Aslam;

```
- Result Grid:** Shows a single row of data with columns: Std_ID, Roll_No, Name, Age, and Mob_No, all containing the value '1001'.
- Toolbar:** Includes icons for Save, Undo, Redo, Copy, Paste, Find, and various export/import options.
- Help:** Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

EXPERIMENT 2

OBJECTIVE : Program to Alter Table.

SOFTWARE USED : MySQL Workbench 8.0 CE

THEORY

ALTER : It is a DDL command. It is used to modify table structure in SQL.

SYNTAX

1. To add a column.

```
Alter table <Table Name>
ADD (new col.name data type(size));
```

2. To drop a column

```
ALTER Table <Table_Name>
DROP column column_name ;
```

3. To modify a column

```
ALTER table table_name
MODIFY column_name new_Datatype (New Size);
```

4. To Rename a Column

```
ALTER table table_name
RENAME column column_name to new_column_name
```

Adding a Column

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'Schemas' section, the 'masooldb' schema is selected. In the central SQL editor window, the following SQL code is visible:

```

1 • USE masooldb;
2 •
3 • ALTER TABLE Masood_Aslam
4 • ADD Email VARCHAR(100);
5 •
6 • Select * from Masood_Aslam;

```

Below the SQL editor is a 'Result Grid' showing the structure of the 'Masood_Aslam' table:

Std_ID	Roll_No	Name	Age	Mob_No	Email
NULL	NULL	NULL	NULL	NULL	NULL

The right-hand sidebar contains tabs for 'Result Grid', 'Form Editor', and 'Field Types'. The 'Result Grid' tab is currently active.

Dropping a Column

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'Schemas' section, the 'masooldb' schema is selected. In the central SQL editor window, the following SQL code is visible:

```

1 • USE masooldb;
2 •
3 • ALTER TABLE Masood_Aslam
4 • DROP COLUMN Age;
5 •
6 • Select * from Masood_Aslam;

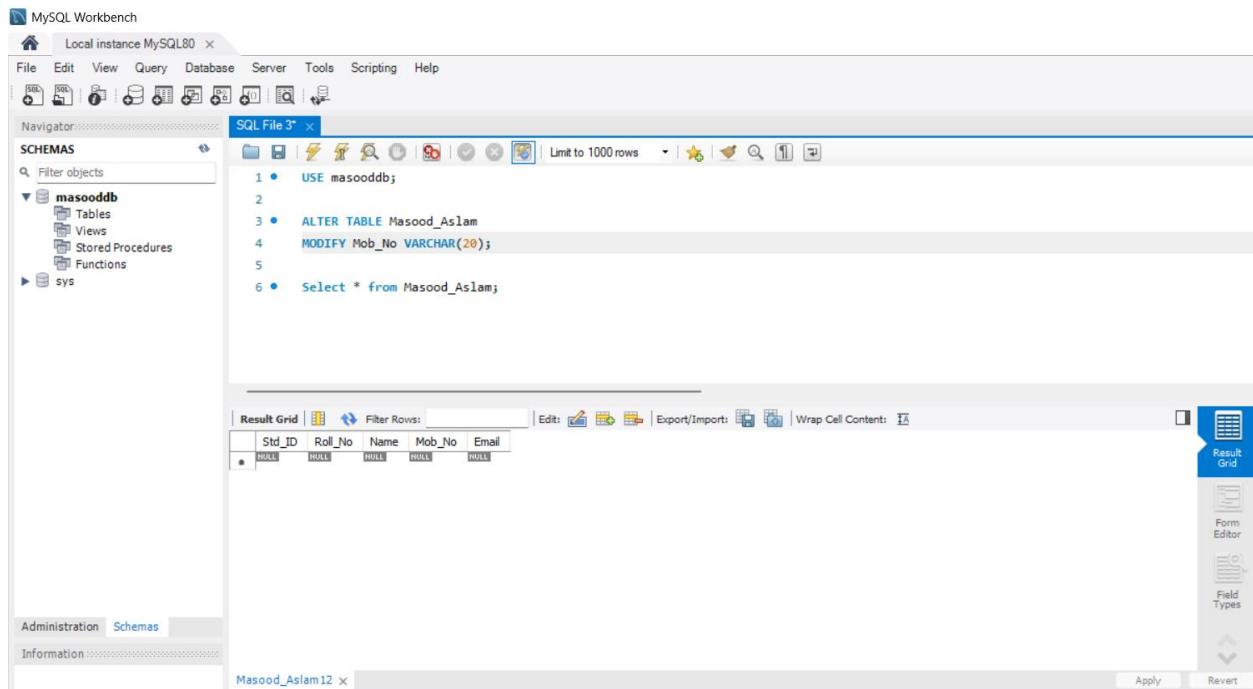
```

Below the SQL editor is a 'Result Grid' showing the structure of the 'Masood_Aslam' table after the 'Age' column was dropped:

Std_ID	Roll_No	Name	Mob_No	Email
NULL	NULL	NULL	NULL	NULL

The right-hand sidebar contains tabs for 'Result Grid', 'Form Editor', and 'Field Types'. The 'Result Grid' tab is currently active.

Modifying a Column



The screenshot shows the MySQL Workbench interface. In the SQL editor, the following SQL code is run:

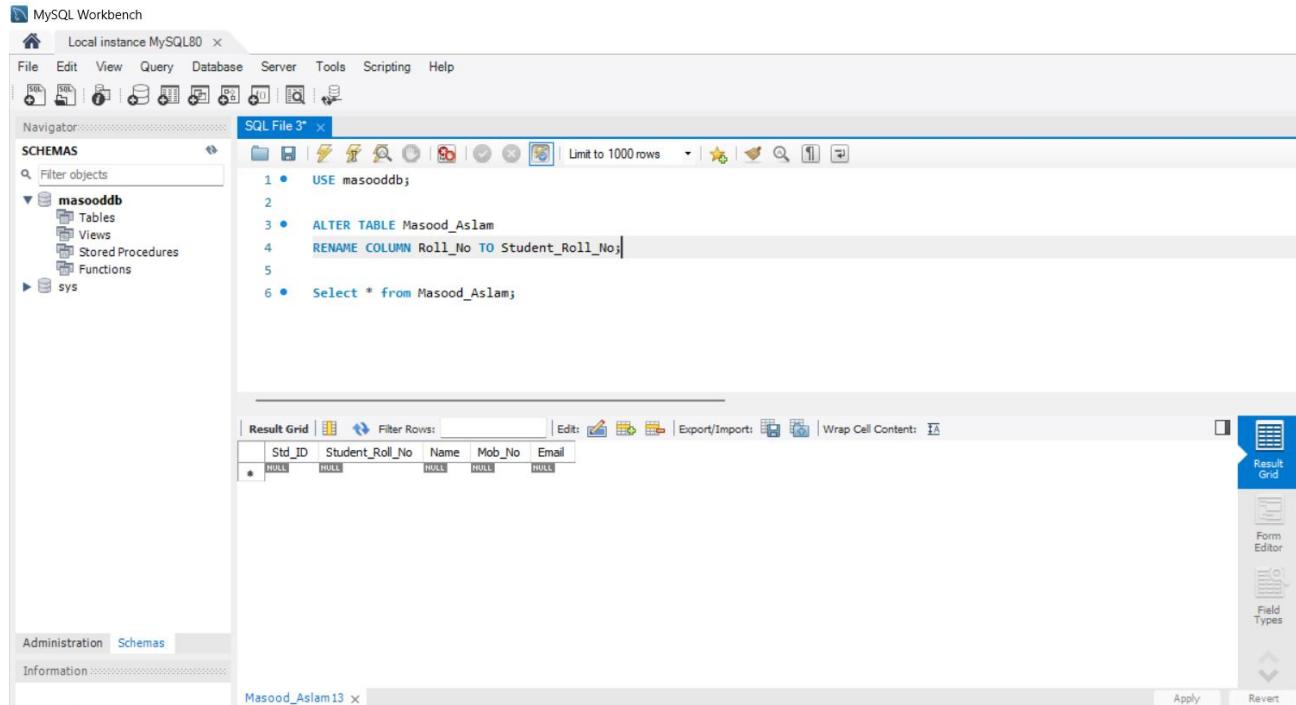
```

1 • USE masoodd;
2
3 • ALTER TABLE Masood_Aslam
4   MODIFY Mob_No VARCHAR(20);
5
6 • Select * from Masood_Aslam;

```

The Result Grid shows the table structure with columns: Std_ID, Roll_No, Name, Mob_No, Email. All columns currently have a length of 5.

Renaming a Column



The screenshot shows the MySQL Workbench interface. In the SQL editor, the following SQL code is run:

```

1 • USE masoodd;
2
3 • ALTER TABLE Masood_Aslam
4   RENAME COLUMN Roll_No TO Student_Roll_No;
5
6 • Select * from Masood_Aslam;

```

The Result Grid shows the table structure with columns: Std_ID, Student_Roll_No, Name, Mob_No, Email. The column 'Roll_No' has been renamed to 'Student_Roll_No'.

EXPERIMENT 3

OBJECTIVE : Program to Insert the Values in a table.

SOFTWARE USED : MySQL Workbench 8.0 CE

THEORY

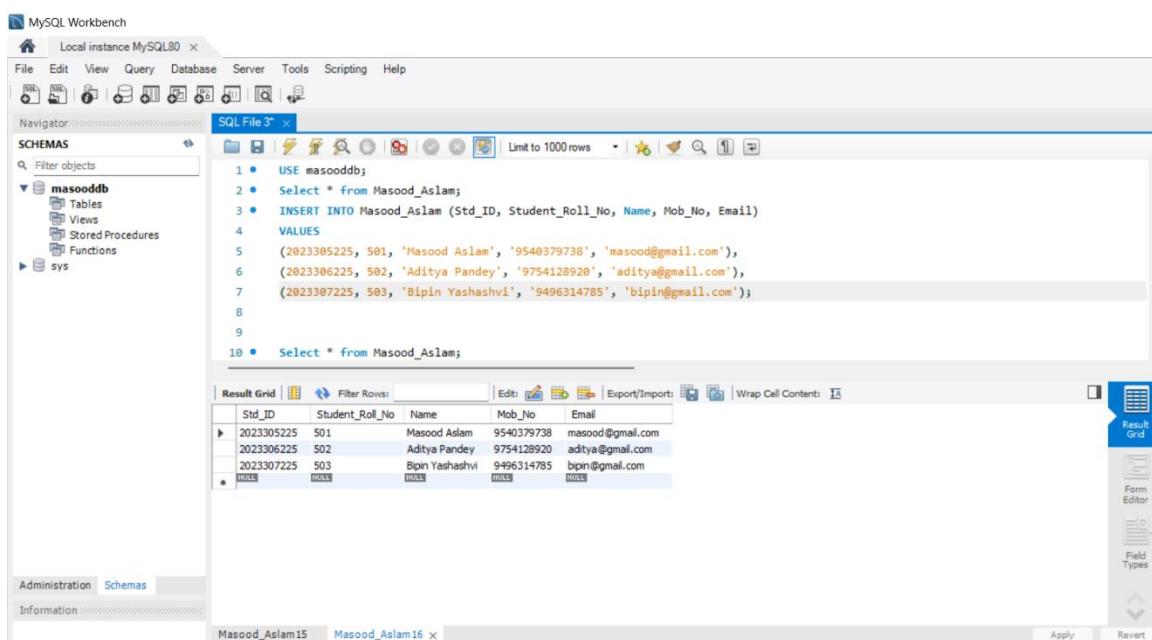
INSERT: It is a DDL command. The insert statement uses the set clause, which specifies the values of a column. You can use more than one set clause per insert statement, and each set clause can set the values in more than one column. Multiple set clauses are not separated by commas. If you specify an optional list of columns, then you can set a value only for a column that is specified in the list of columns to be inserted and when character value insert then apply (' ') example ('value') otherwise not apply.

SYNTAX

1. To add values in to table

Insert into table_name
Values (value,.....,value);

Inserting values in a table



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

```

USE masoodeb;
Select * from Masood_Aslam;
INSERT INTO Masood_Aslam (Std_ID, Student_Roll_No, Name, Mob_No, Email)
VALUES
(2023305225, 501, 'Masood Aslam', '9540379738', 'masood@gmail.com'),
(2023306225, 502, 'Aditya Pandey', '9754128920', 'aditya@gmail.com'),
(2023307225, 503, 'Bipin Yashashvi', '9496314785', 'bipin@gmail.com');

Select * from Masood_Aslam;
  
```

The Result Grid shows the following data:

Std_ID	Student_Roll_No	Name	Mob_No	Email
2023305225	501	Masood Aslam	9540379738	masood@gmail.com
2023306225	502	Aditya Pandey	9754128920	aditya@gmail.com
2023307225	503	Bipin Yashashvi	9496314785	bipin@gmail.com
*	HULL		HULL	HULL

EXPERIMENT 4

OBJECTIVE : Program to View & Filter data in table.

SOFTWARE USED : MySQL Workbench 8.0 CE

THEORY

Once data has been inserted into a table, the next most logical operation would be to view what has been inserted. While viewing data from a table, it is rare that all the data from table will be required each time. Hence, SQL must give us a method of filtering out data that is not data required.

SYNTAX

1. All Rows and All Columns

Select * from table_name;

2. Filtering Table Data

a. Select Columns & all Rows

select column_name1, column_name2
from table_name;

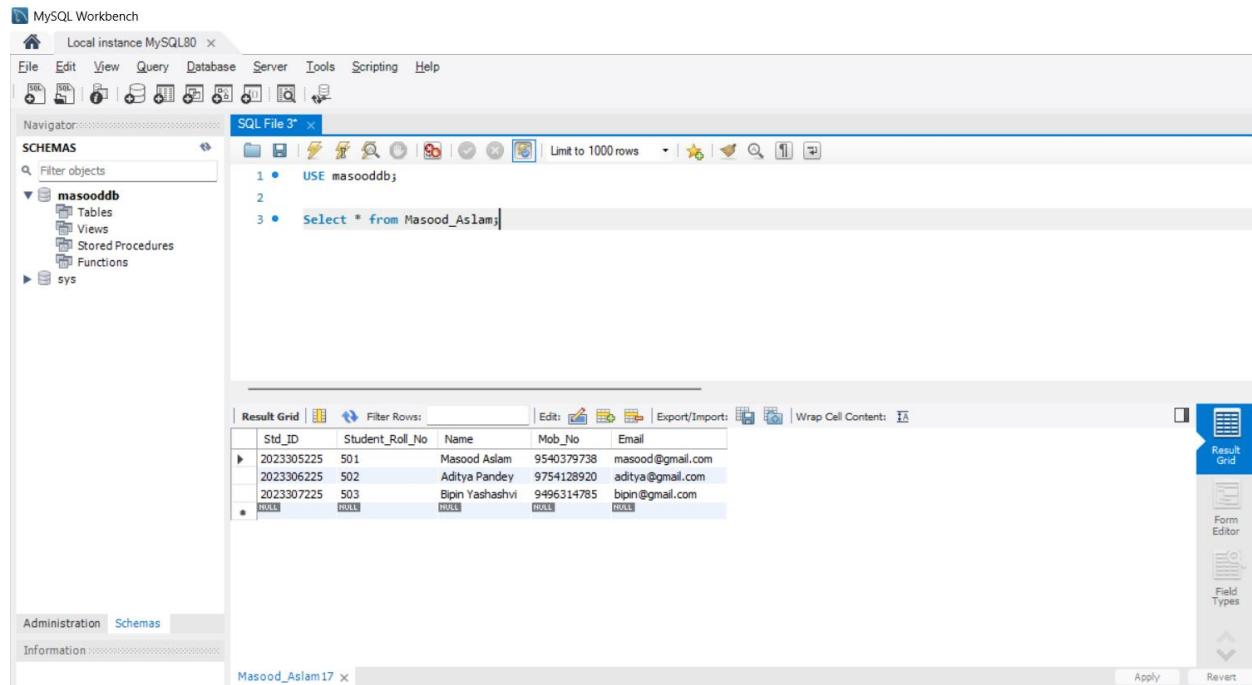
b. Select Rows and all Columns

select * from table_name
where condition;

c. Select Rows and select Columns

select column_name1, column_name2
from table_name
where condition;

All Rows and All Columns



MySQL Workbench - Local instance MySQL80

SQL File 3*

```

1 • USE masoodeb;
2
3 • Select * from Masood_Aslam;
  
```

Result Grid

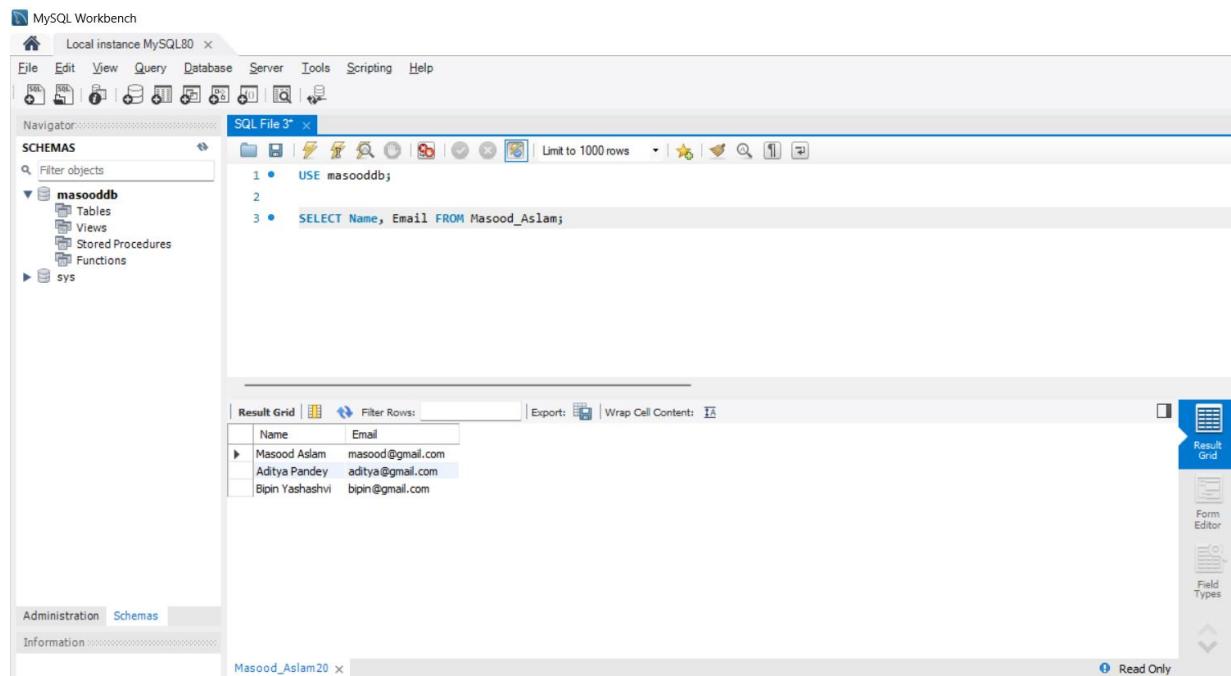
Std_ID	Student_Roll_No	Name	Mob_No	Email
2023305225	501	Masood Aslam	9540379738	masood@gmail.com
2023306225	502	Aditya Pandey	9754128920	aditya@gmail.com
2023307225	503	Bipin Yashashvi	9496314785	bipin@gmail.com
NULL	NULL	NULL	NULL	NULL
•	•	•	•	•

Administration Schemas

Information

Masood_Aslam17

Select Columns & all Rows



MySQL Workbench - Local instance MySQL80

SQL File 3*

```

1 • USE masoodeb;
2
3 • SELECT Name, Email FROM Masood_Aslam;
  
```

Result Grid

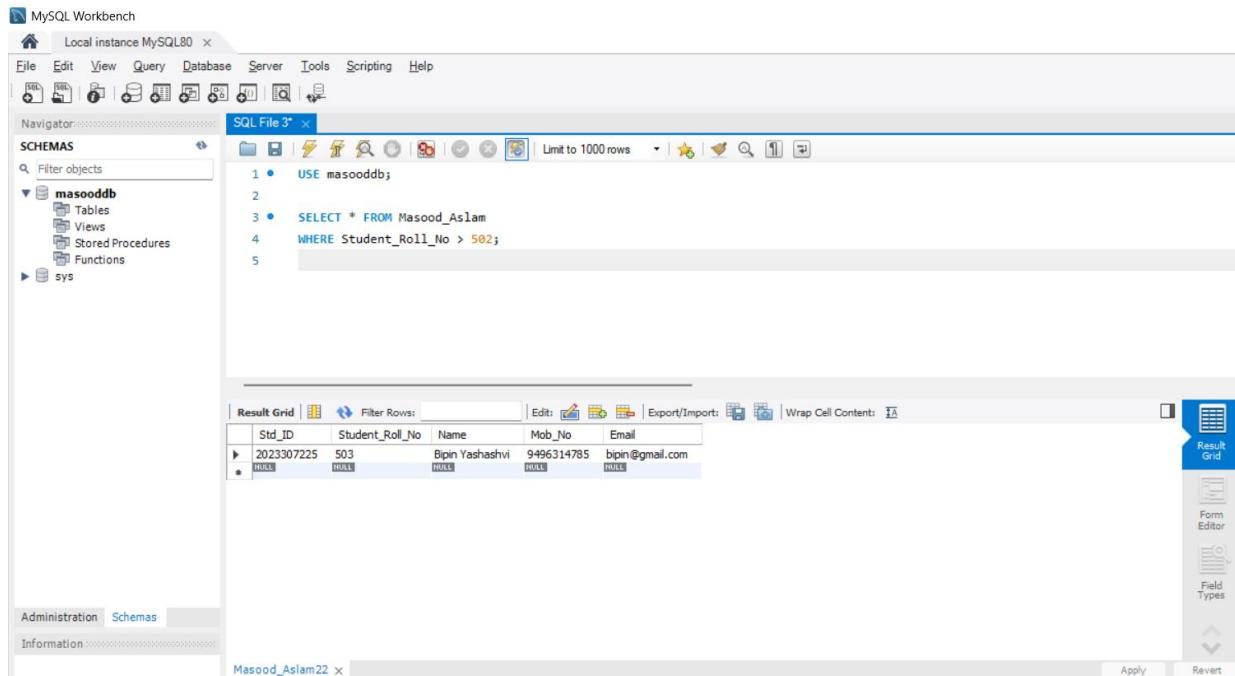
Name	Email
Masood Aslam	masood@gmail.com
Aditya Pandey	aditya@gmail.com
Bipin Yashashvi	bipin@gmail.com

Administration Schemas

Information

Masood_Aslam20

Select Rows and all Columns



The screenshot shows the MySQL Workbench interface. In the SQL editor, a query is written to select all columns from the 'Masood_Aslam' table where the 'Student_Roll_No' is greater than 502:

```

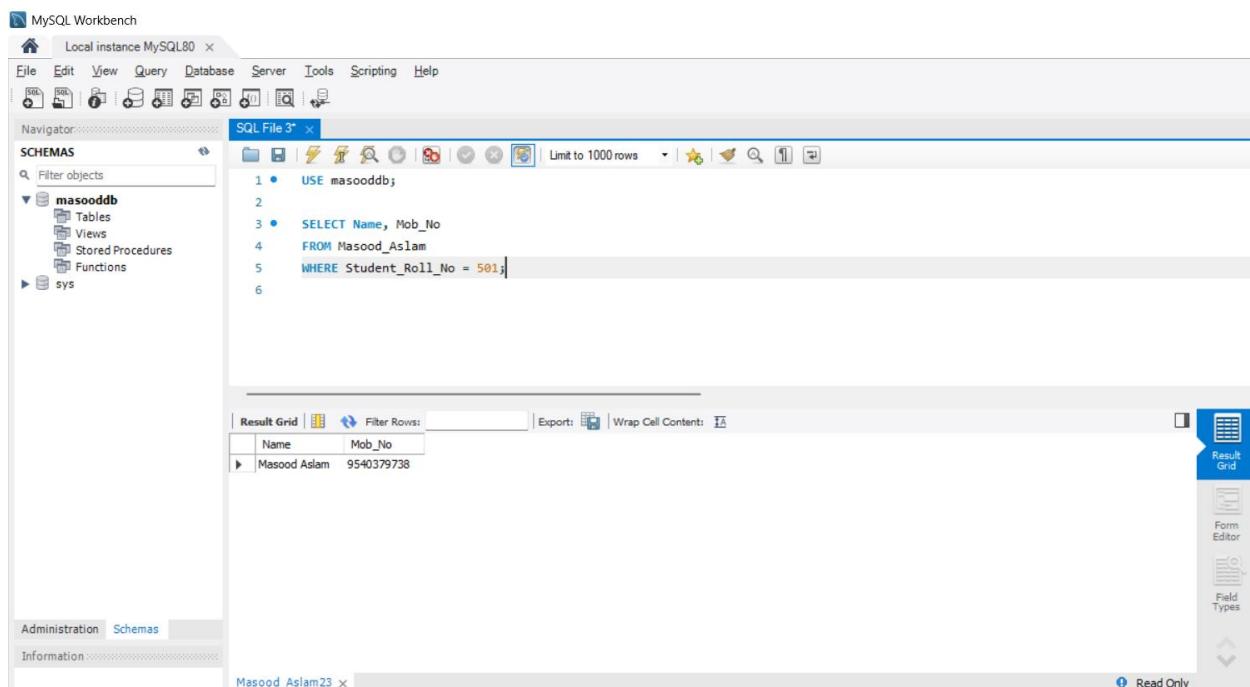
1 • USE masoodebb;
2
3 • SELECT * FROM Masood_Aslam
4 WHERE Student_Roll_No > 502;
5

```

The result grid displays one row of data:

Std_ID	Student_Roll_No	Name	Mob_No	Email
202307225	503	Bipin Yashashi	9496314785	bipin@gmail.com

Select Rows and select Columns



The screenshot shows the MySQL Workbench interface. A query is run to select the 'Name' and 'Mob_No' columns from the 'Masood_Aslam' table where the 'Student_Roll_No' is 501:

```

1 • USE masoodebb;
2
3 • SELECT Name, Mob_No
4   FROM Masood_Aslam
5 WHERE Student_Roll_No = 501;
6

```

The result grid displays one row of data:

Name	Mob_No
Masood Aslam	9540379738

EXPERIMENT 5

OBJECTIVE : Program to delete record of a table.

SOFTWARE USED : Microsoft SQL Server 2008 R2

THEORY

The DELETE statement is used to delete rows in a table.

SYNTAX

```
DELETE FROM table_name
WHERE some_column=some_value;
```

The screenshot shows the MySQL Workbench interface. In the top-left corner, the title bar reads "MySQL Workbench Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. On the left, the Navigator pane shows the "SCHEMAS" section with "masooddb" selected, displaying Tables, Views, Stored Procedures, and Functions. The main area has a tab titled "SQL File 3" containing the following SQL code:

```

1 • USE masooddb;
2 • DELETE FROM Masood_Aslam
3 • WHERE Email = 'bipin@gmail.com';
4 • select * from Masood_Aslam;
5

```

Below the SQL editor is the "Result Grid" pane, which displays a table with the following data:

Std_ID	Student_Roll_No	Name	Mob_No	Email
2023305225	501	Masood Aslam	9540379738	masood@gmail.com
2023306225	502	Aditya Pandey	9754128920	aditya@gmail.com
HULL	HULL	HULL	HULL	HULL

On the right side of the interface, there are three tabs: "Result Grid" (selected), "Form Editor", and "Field Types". At the bottom, there are "Apply" and "Revert" buttons.

EXPERIMENT 6

OBJECTIVE : Program to update the content of a table.

SOFTWARE USED : Microsoft SQL Server 2008 R2

THEORY

The UPDATE statement is used to update existing records in a table.

SYNTAX

```
UPDATE table_name
SET column1=value1,column2=value2...
WHERE some_column=some_value;
```

Before Updates

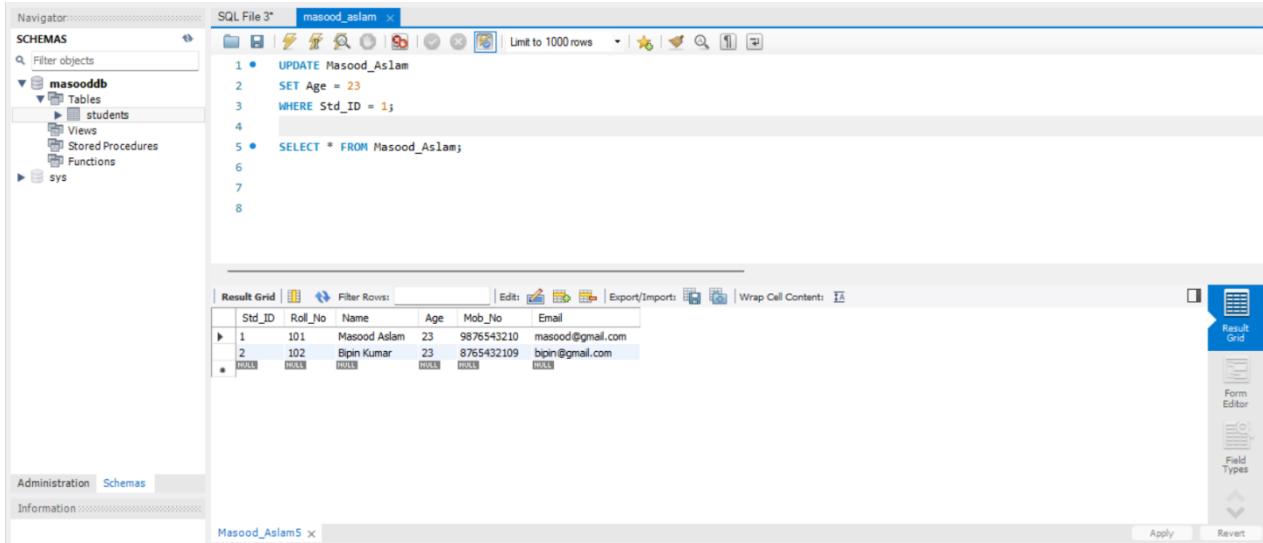
The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the database structure with the schema **masooddbs** expanded, revealing **Tables** (including **students**), **Views**, **Stored Procedures**, and **Functions**.
- SQL File 3*** window: Contains the T-SQL command:

```
1
2 *  SELECT * FROM Masood_Aslam;
3
4
5
```
- Result Grid:** Displays the initial data from the **students** table:

Std_ID	Roll_No	Name	Age	Mob_No	Email
1	101	Masood Aslam	22	9876543210	masood@gmail.com
2	102	Bipin Kumar	23	8765432109	bipin@gmail.com
HULL	HULL	HULL	HULL	HULL	HULL
- Toolbars and Status Bar:** Standard SSMS toolbars and status bar at the bottom.

Updating a Single Column



The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the schema 'masooldb' selected. The middle pane shows a SQL file named 'masood_aslam' containing the following code:

```

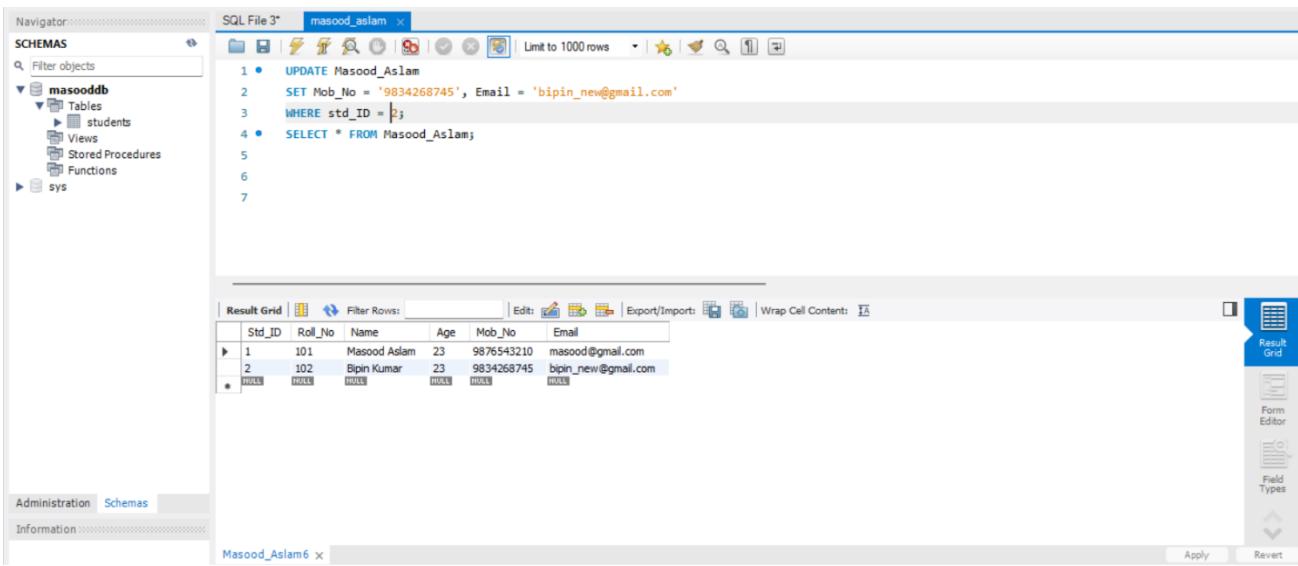
1 • UPDATE Masood_Aslam
2     SET Age = 23
3     WHERE Std_ID = 1;
4
5 • SELECT * FROM Masood_Aslam;
6
7
8

```

The bottom pane shows the results of the query in a grid:

Std_ID	Roll_No	Name	Age	Mob_No	Email
1	101	Masood Aslam	23	9876543210	masood@gmail.com
2	102	Bipin Kumar	23	8765432109	bipin@gmail.com
*	NULL	NULL	NULL	NULL	NULL

Updating Multiple Columns



The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the schema 'masooldb' selected. The middle pane shows a SQL file named 'masood_aslam' containing the following code:

```

1 • UPDATE Masood_Aslam
2     SET Mob_No = '9834268745', Email = 'bipin_new@gmail.com'
3     WHERE std_ID = 2;
4 • SELECT * FROM Masood_Aslam;
5
6
7

```

The bottom pane shows the results of the query in a grid:

Std_ID	Roll_No	Name	Age	Mob_No	Email
1	101	Masood Aslam	23	9876543210	masood@gmail.com
2	102	Bipin Kumar	23	9834268745	bipin_new@gmail.com
*	NULL	NULL	NULL	NULL	NULL

EXPERIMENT 7

OBJECTIVE : Write a program to sort the data of a table.

SOFTWARE USED : Microsoft SQL Server 2008 R2

THEORY

The ORDER BY keyword is used to sort the result-set by one or more columns. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

SYNTAX

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

Before Sorting:

Std_ID	Roll_No	Name	Age	Mob_No	Email
1	101	Masood Aslam	23	9876543210	masood@gmail.com
2	102	Bipin Kumar	23	9834268745	bipin_new@gmail.com
3	103	Aditya Pandey	22	7798234512	aditya@gmail.com
4	104	Lakshay Sharma	23	9876341287	sharma@gmail.com

Sorting Data in Ascending Order (Default):

The screenshot shows the SQL Server Management Studio interface. In the top-left pane, the Navigator displays the schema 'masooddbs' with its tables, views, stored procedures, and functions. The central pane contains a SQL query window titled 'SQL File 3*' with the following code:

```

1 • SELECT * FROM Masood_Aslam
2 ORDER BY Name;

```

The bottom-right pane shows the 'Result Grid' with the following data:

Std_ID	Roll_No	Name	Age	Mob_No	Email
3	103	Aditya Pandey	22	7798234512	aditya@gmail.com
2	102	Bipin Kumar	23	9834268745	bipin_new@gmail.com
4	104	Lakshay Sharma	23	9876341287	sharma@gmail.com
1	101	Masood Aslam	23	9876543210	masood@gmail.com
*	HULL	HULL	HULL	HULL	HULL

Sorting Data in Descending Order:

The screenshot shows the SQL Server Management Studio interface. In the top-left pane, the Navigator displays the schema 'masooddbs' with its tables, views, stored procedures, and functions. The central pane contains a SQL query window titled 'SQL File 3*' with the following code:

```

1 • SELECT * FROM Masood_Aslam
2 ORDER BY Age DESC;

```

The bottom-right pane shows the 'Result Grid' with the following data:

Std_ID	Roll_No	Name	Age	Mob_No	Email
1	101	Masood Aslam	23	9876543210	masood@gmail.com
2	102	Bipin Kumar	23	9834268745	bipin_new@gmail.com
4	104	Lakshay Sharma	23	9876341287	sharma@gmail.com
3	103	Aditya Pandey	22	7798234512	aditya@gmail.com
*	HULL	HULL	HULL	HULL	HULL

Sorting by Multiple Columns:

The screenshot shows the SQL Server Management Studio interface. In the top-left pane, the Navigator displays the schema 'masooldb' with its tables, views, stored procedures, and functions. The central area contains a SQL editor window titled 'SQL File 3*' with the following query:

```
1 • SELECT * FROM Masood_Aslam
2 ORDER BY Age ASC, Name DESC;
```

Below the SQL editor is a 'Result Grid' window showing the data from the 'students' table. The columns are Std_ID, Roll_No, Name, Age, Mob_No, and Email. The rows are sorted by Age in ascending order and Name in descending order. The data is as follows:

	Std_ID	Roll_No	Name	Age	Mob_No	Email
▶	3	103	Aditya Pandey	22	7798234512	aditya@gmail.com
1	101		Masood Aslam	23	9876543210	masood@gmail.com
4	104		Lakshay Sharma	23	9876341287	sharma@gmail.com
2	102		Bipin Kumar	23	9834268745	bipin_new@gmail.com
*	HULL	HULL	HULL	HULL	HULL	HULL

Sorting Specific Columns:

The screenshot shows the SQL Server Management Studio interface. In the top-left pane, the Navigator displays the schema 'masooldb' with its tables, views, stored procedures, and functions. The central area contains a SQL editor window titled 'SQL File 3*' with the following query:

```
1 • SELECT Name, Age FROM Masood_Aslam
2 ORDER BY Age DESC;
```

Below the SQL editor is a 'Result Grid' window showing the data from the 'students' table, specifically selecting the 'Name' and 'Age' columns. The rows are sorted by 'Age' in descending order. The data is as follows:

	Name	Age
▶	Masood Aslam	23
	Bipin Kumar	23
	Lakshay Sharma	23
	Aditya Pandey	22

EXPERIMENT 8

OBJECTIVE : Write a program to implement set functioning SQL.

SOFTWARE USED : Microsoft SQL Server 2008 R2

THEORY

SQL supports few operations to be performed on table data.

UNION

Union is used to combine the results of two or more select statements.

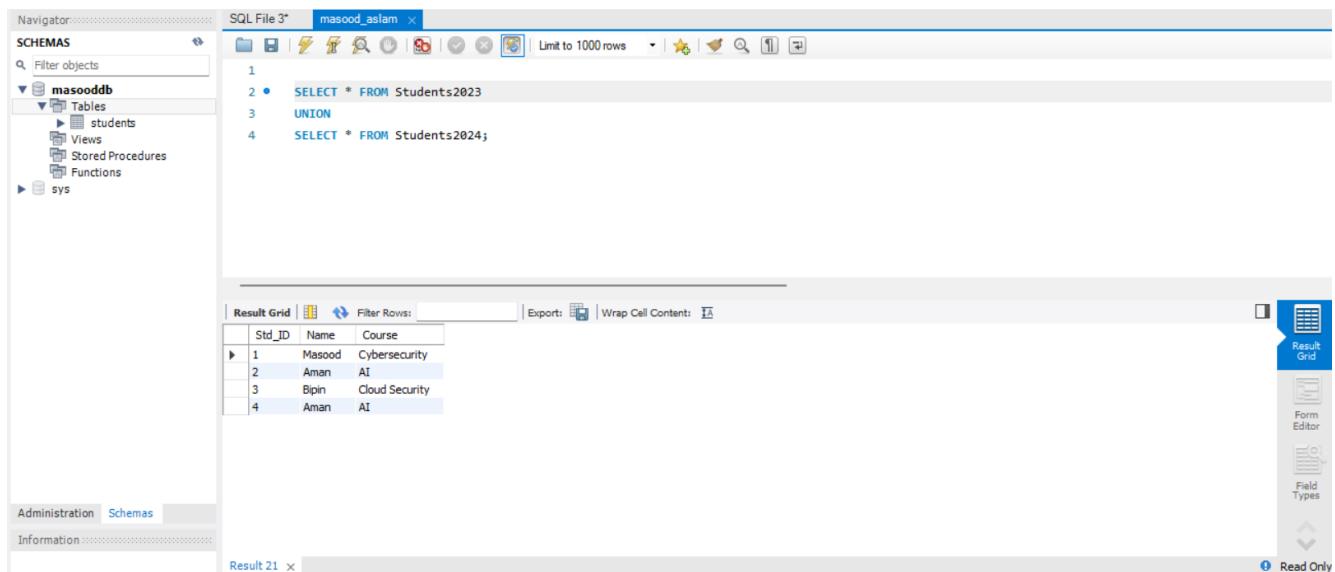
SYNTAX

SELECT * FROM first_table

UNION

SELECT * FROM Second_table;

Using UNION to Merge Both Tables:



The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the 'masooldb' schema selected, showing tables like 'students', 'Views', 'Stored Procedures', and 'Functions'. The right pane shows a query window titled 'SQL File 3' with the following code:

```

1
2 • SELECT * FROM Students2023
3 UNION
4 SELECT * FROM Students2024;

```

Below the code, the 'Result Grid' shows the merged data from both tables:

Std_ID	Name	Course
1	Masood	Cybersecurity
2	Aman	AI
3	Bipin	Cloud Security
4	Aman	AI

Using UNION with Specific Columns:

The screenshot shows a SQL database interface with the following details:

- Navigator:** Shows the schema structure with **masooldb** selected, containing **Tables** (students), **Views**, **Stored Procedures**, and **Functions**.
- SQL File 3*** tab: The query entered is:

```
1
2 •  SELECT Name, Course FROM Students2023
3 UNION
4  SELECT Name, Course FROM Students2024;
5
```
- Result Grid:** Displays the query results in a table format:

Name	Course
Masood	Cybersecurity
Aman	AI
Bipin	Cloud Security
- Information:** Shows "Result 26" and a "Read Only" status.
- Toolbar:** Includes standard database icons for file operations, search, and refresh.

EXPERIMENT 9

OBJECTIVE : Write a program to implement JOIN operation on two tables.

SOFTWARE USED : Microsoft SQL Server 2008 R2

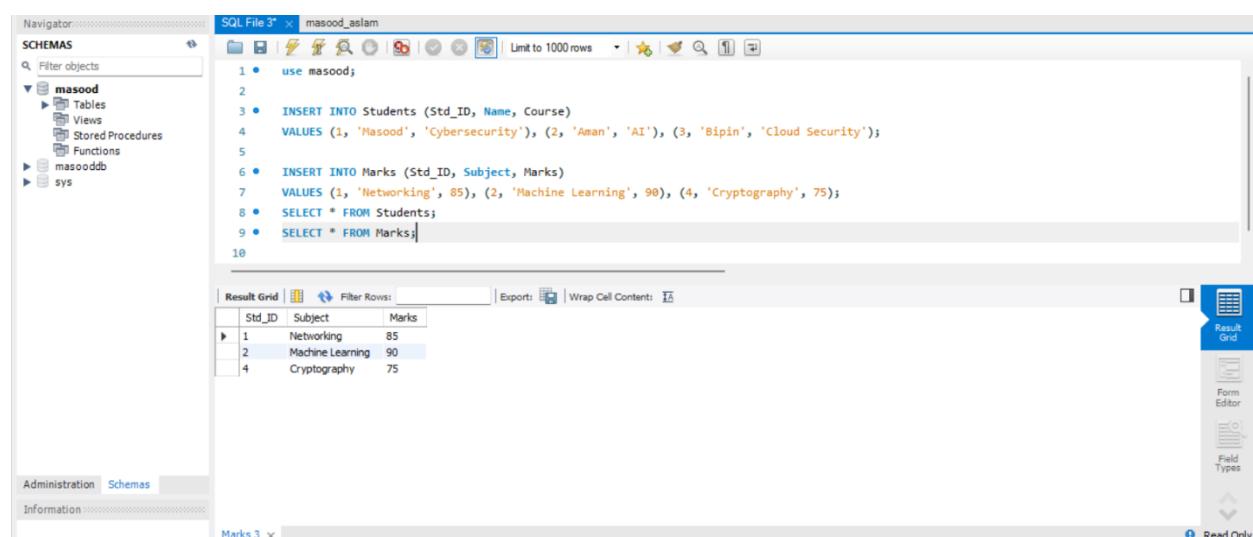
THEORY

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them. The most common type of join is: SQL INNER JOIN (simple join). An SQL INNER JOIN returns all rows from multiple tables where the join condition is met.

SYNTAX

```
SELECT * FROM first_table,Second_table
WHERE first_table.common_attribute=second_table.common_attribute;
```

Inserting Sample Data



The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the database structure with Schemas, Tables, Views, Stored Procedures, and Functions under the 'masood' schema.
- SQL File 3:** The current query window contains the following SQL code:


```
1 • use masood;
2
3 • INSERT INTO Students (Std_ID, Name, Course)
4   VALUES (1, 'Masood', 'Cybersecurity'), (2, 'Aman', 'AI'), (3, 'Bipin', 'Cloud Security');
5
6 • INSERT INTO Marks (Std_ID, Subject, Marks)
7   VALUES (1, 'Networking', 85), (2, 'Machine Learning', 90), (4, 'Cryptography', 75);
8 • SELECT * FROM Students;
9 • SELECT * FROM Marks;
```
- Result Grid:** Displays the data inserted into the 'Marks' table:

Std_ID	Subject	Marks
1	Networking	85
2	Machine Learning	90
4	Cryptography	75
- Status Bar:** Shows 'Read Only' status.

INNER JOIN (Common Matching Records Only)

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the 'masood' schema selected. The main pane contains the following SQL code:

```

1 • use masood;
2 • SELECT Students.Std_ID, Students.Name, Students.Course, Marks.Subject, Marks.Marks
3 FROM Students
4 INNER JOIN Marks
5 ON Students.Std_ID = Marks.Std_ID;
6
7
8

```

The results grid shows the following data:

Std_ID	Name	Course	Subject	Marks
1	Masood	Cybersecurity	Networking	85
2	Aman	AI	Machine Learning	90

LEFT JOIN (All Students + Marks if Available)

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the 'masood' schema selected. The main pane contains the following SQL code:

```

1 • use masood;
2 • SELECT Students.Std_ID, Students.Name, Students.Course, Marks.Subject, Marks.Marks
3 FROM Students
4 LEFT JOIN Marks
5 ON Students.Std_ID = Marks.Std_ID;
6
7
8
9

```

The results grid shows the following data:

Std_ID	Name	Course	Subject	Marks
1	Masood	Cybersecurity	Networking	85
2	Aman	AI	Machine Learning	90
3	Bipin	Cloud Security	NULL	NULL

RIGHT JOIN (All Marks + Students if Available)

The screenshot shows the SQL Server Management Studio interface. The top pane displays a T-SQL script for a RIGHT JOIN:

```

1 • use masood;
2 • SELECT Students.Std_ID, Students.Name, Students.Course, Marks.Subject, Marks.Marks
3   FROM Students
4   RIGHT JOIN Marks
5     ON Students.Std_ID = Marks.Std_ID;
6
7
8
9

```

The bottom pane shows the result grid with the following data:

Std_ID	Name	Course	Subject	Marks
1	Masood	Cybersecurity	Networking	85
2	Aman	AI	Machine Learning	90
			Cryptography	75

FULL OUTER JOIN (All Data, Even If No Match)

The screenshot shows the SQL Server Management Studio interface. The top pane displays a T-SQL script for a FULL OUTER JOIN, using UNION to combine the results of a LEFT JOIN and a RIGHT JOIN:

```

1 • use masood;
2 • SELECT Students.Std_ID, Students.Name, Students.Course, Marks.Subject, Marks.Marks
3   FROM Students
4   LEFT JOIN Marks ON Students.Std_ID = Marks.Std_ID
5
6 UNION
7
8 • SELECT Students.Std_ID, Students.Name, Students.Course, Marks.Subject, Marks.Marks
9   FROM Students
10  RIGHT JOIN Marks ON Students.Std_ID = Marks.Std_ID;

```

The bottom pane shows the result grid with the following data:

Std_ID	Name	Course	Subject	Marks
1	Masood	Cybersecurity	Networking	85
2	Aman	AI	Machine Learning	90
3	Bipin	Cloud Security		
			Cryptography	75

EXPERIMENT 10

OBJECTIVE : Write a program to implement Aggregate functions using SQL.

SOFTWARE USED : Microsoft SQL Server 2008 R2

THEORY

The SQL aggregate functions, as their title suggests are used to retrieve minimum and maximum values from a column, to sum values in a column, to get the average of a column values, or to simply count a number of records according to a search condition.

SYNTAX

The most commonly used SQL aggregate function is the COUNT function;;

SELECT COUNT(*) FROM table_name WHERE condition;

We can select minimum and maximum from a Table as follows:

SELECT MIN(Column_name) FROM table_name;

SELECT MAX(Column_name)FROM table_name;

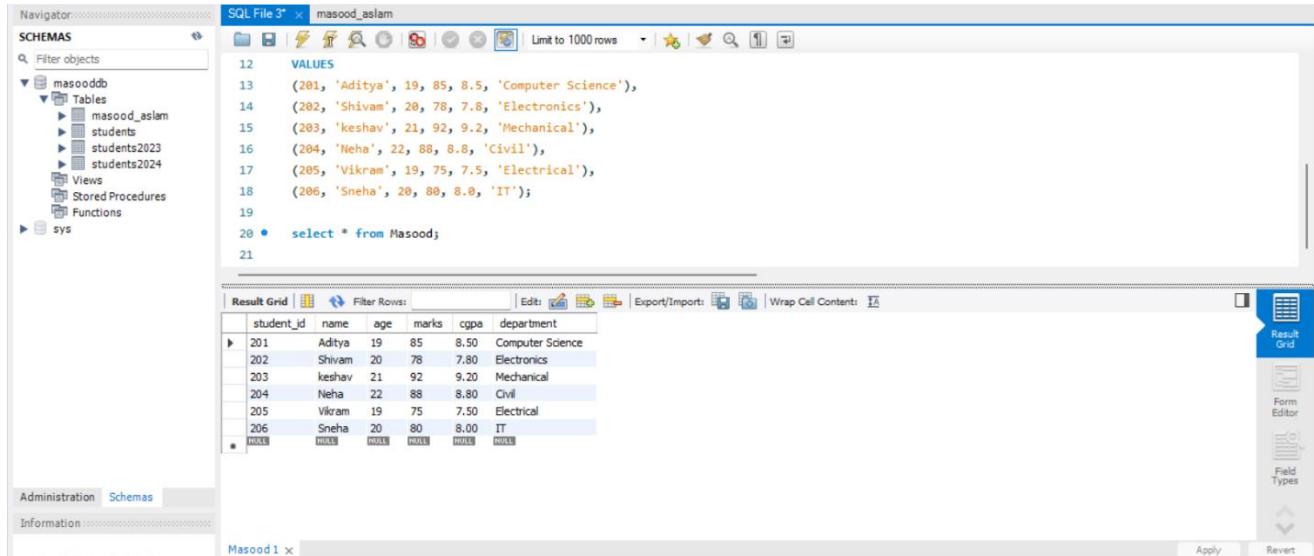
We can also select the average from a Table as follows:

SELECT AVG(Column_name)FROM Table_name;

We can use the SUM aggregate function to get the sum of values in a certain column:

SELECT SUM(Column_name) FROM table_name;

Inserting Sample Data



The screenshot shows the SQL File 3 window with the following code:

```

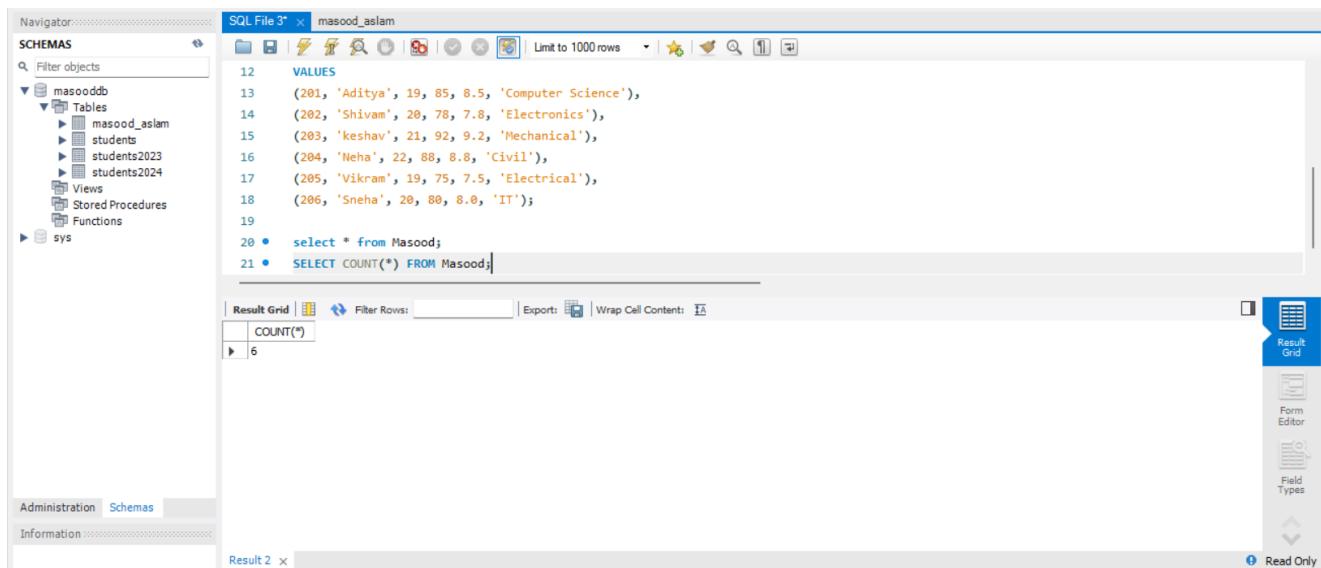
12     VALUES
13     (201, 'Aditya', 19, 85, 8.5, 'Computer Science'),
14     (202, 'Shivam', 20, 78, 7.8, 'Electronics'),
15     (203, 'keshav', 21, 92, 9.2, 'Mechanical'),
16     (204, 'Neha', 22, 88, 8.8, 'Civil'),
17     (205, 'Vikram', 19, 75, 7.5, 'Electrical'),
18     (206, 'Sneha', 20, 80, 8.0, 'IT');
19
20 • select * from Masood;
21

```

The Result Grid shows the inserted data:

student_id	name	age	marks	cgpa	department
201	Aditya	19	85	8.50	Computer Science
202	Shivam	20	78	7.80	Electronics
203	keshav	21	92	9.20	Mechanical
204	Neha	22	88	8.80	Civil
205	Vikram	19	75	7.50	Electrical
206	Sneha	20	80	8.00	IT
NULL	NULL	NULL	NULL	NULL	NULL

Count Total Rows



The screenshot shows the SQL File 3 window with the following code:

```

12     VALUES
13     (201, 'Aditya', 19, 85, 8.5, 'Computer Science'),
14     (202, 'Shivam', 20, 78, 7.8, 'Electronics'),
15     (203, 'keshav', 21, 92, 9.2, 'Mechanical'),
16     (204, 'Neha', 22, 88, 8.8, 'Civil'),
17     (205, 'Vikram', 19, 75, 7.5, 'Electrical'),
18     (206, 'Sneha', 20, 80, 8.0, 'IT');
19
20 • select * from Masood;
21 • SELECT COUNT(*) FROM Masood;

```

The Result Grid shows the count of rows:

COUNT(*)
6

Find Sum of Values

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the Schemas section expanded, showing the masoodddb schema with its tables (masood_aslam, students, students2023, students2024). The right pane shows the results of a SQL file named 'masood_aslam'. The code in the editor is:

```

13  (201, 'Aditya', 19, 85, 8.5, 'Computer Science'),
14  (202, 'Shivam', 20, 78, 7.8, 'Electronics'),
15  (203, 'Keshav', 21, 92, 9.2, 'Mechanical'),
16  (204, 'Neha', 22, 88, 8.8, 'Civil'),
17  (205, 'Vikram', 19, 75, 7.5, 'Electrical'),
18  (206, 'Sneha', 20, 80, 8.0, 'IT')

19
20 • select * from Masood;
21 • SELECT COUNT(*) FROM Masood;
22 • SELECT SUM(MARKS) FROM Masood;

```

The Result Grid shows the output of the last query:

SUM(MARKS)
498

Calculate Average Value

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the Schemas section expanded, showing the masoodddb schema with its tables (masood_aslam, students, students2023, students2024). The right pane shows the results of a SQL file named 'masood_aslam'. The code in the editor is:

```

14  (202, 'Shivam', 20, 78, 7.8, 'Electronics'),
15  (203, 'Keshav', 21, 92, 9.2, 'Mechanical'),
16  (204, 'Neha', 22, 88, 8.8, 'Civil'),
17  (205, 'Vikram', 19, 75, 7.5, 'Electrical'),
18  (206, 'Sneha', 20, 80, 8.0, 'IT')

19
20 • select * from Masood;
21 • SELECT COUNT(*) FROM Masood;
22 • SELECT SUM(MARKS) FROM Masood;
23 • SELECT AVG(CGPA) FROM Masood;

```

The Result Grid shows the output of the last query:

AVG(CGPA)
8.300000

Find Minimum & Maximum Values

Schemas

```

15  (203, 'Keshav', 21, 92, 9.2, 'Mechanical'),
16  (204, 'Neha', 22, 88, 8.8, 'Civil'),
17  (205, 'Vikram', 19, 75, 7.5, 'Electrical'),
18  (206, 'Sneha', 20, 80, 8.0, 'IT');

20 • select * from Masood;
21 • SELECT COUNT(*) FROM Masood;
22 • SELECT SUM(MARKS) FROM Masood;
23 • SELECT AVG(CGPA) FROM Masood;
24 • SELECT MAX(MARKS) FROM Masood;

```

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

MAX(MARKS)
92

Result 5 x Read Only

Schemas

```

16  (204, 'Neha', 22, 88, 8.8, 'Civil'),
17  (205, 'Vikram', 19, 75, 7.5, 'Electrical'),
18  (206, 'Sneha', 20, 80, 8.0, 'IT');

20 • select * from Masood;
21 • SELECT COUNT(*) FROM Masood;
22 • SELECT SUM(MARKS) FROM Masood;
23 • SELECT AVG(CGPA) FROM Masood;
24 • SELECT MAX(MARKS) FROM Masood;
25 • SELECT MIN(CGPA) FROM Masood;

```

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

MIN(CGPA)
7.5

Result 6 x Read Only

EXPERIMENT 13

OBJECTIVE : Write a program to implement ORACLE functions in SQL.

SOFTWARE USED : MySQL Workbench 8.0 CE

THEORY

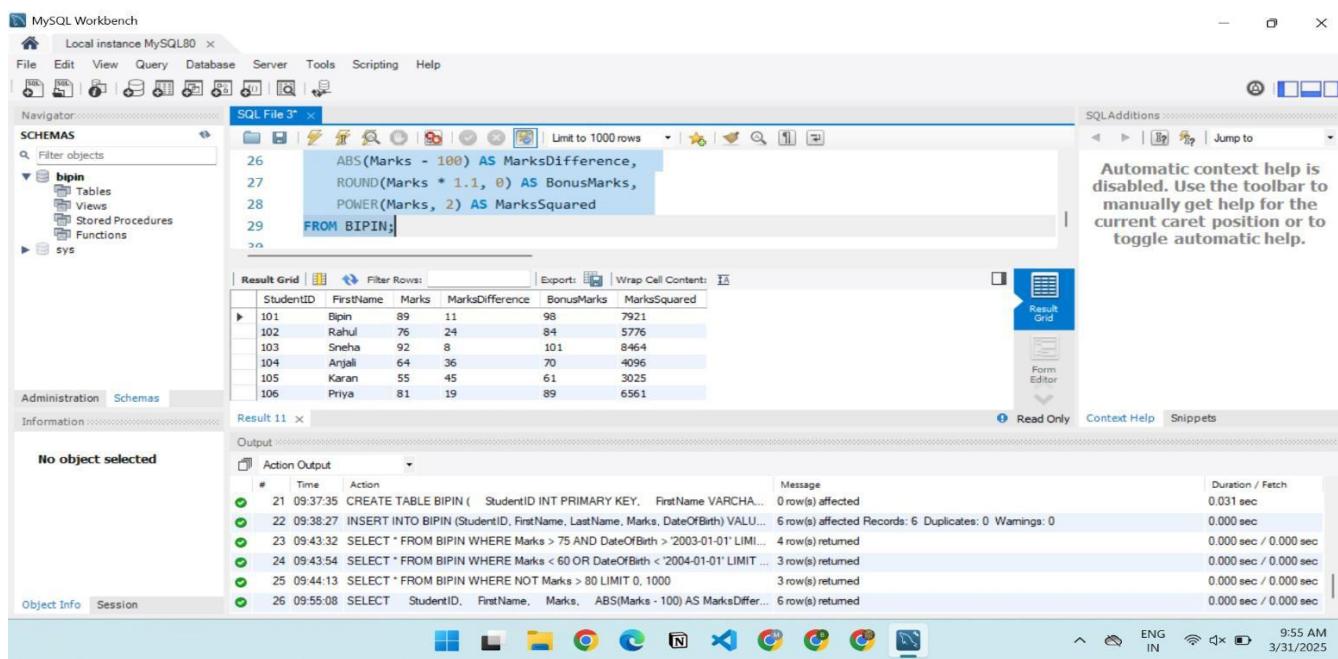
There are four types of single row functions:

1. **NUMERIC FUNCTIONS:** These are functions that accept numeric input and return numeric values.
 2. **CHARACTER FUNCTIONS:** These are functions that accept character input and can return both character and number values.
 3. **DATE FUNCTIONS:** These are that take values that are of data type DATE as input and return values of data type DATE.
 4. **CONVERSION FUNCTIONS:** These are functions that help us to convert a value in one form to another datatype.

DUAL TABLE

This is a single row and single column dummy table provided by oracle. This is used to perform mathematical calculations without using a table.

NUMERIC FUNCTION



CHARACTER FUNCTION

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```

36    SUBSTR.FirstName, 1, 3) AS FirstThreeLetters,
37    CONCAT.FirstName, ' ', LastName AS FullName
38
39  FROM BIPIN;

```

The results grid displays the following data:

FirstName	UpperCaseName	LowerCaseName	NameLength	FirstThreeLetters	FullName
Bipin	BIPIN	yashasvi	5	Bip	Bipin Yashasvi
Rahul	RAHUL	verma	5	Rah	Rahul Verma
Sneha	SNEHA	gupta	5	Sne	Sneha Gupta
Anjali	ANJALI	sharma	6	Anj	Anjali Sharma
Karan	KARAN	singh	5	Kar	Karan Singh
Priya	PRIYA	mishra	5	Pri	Priya Mishra

The status bar at the bottom right shows the date and time: 9:55 AM 3/31/2025.

DATE FUNCTION

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```

45    DATE_ADD(DateOfBirth, INTERVAL 6 MONTH) AS SixMonthsLater,
46    TIMESTAMPDIFF(MONTH, DateOfBirth, CURDATE()) AS MonthsSinceBirth,
47    DATE_FORMAT(CURDATE(), '%Y-01-01') AS StartOfYear
48
49

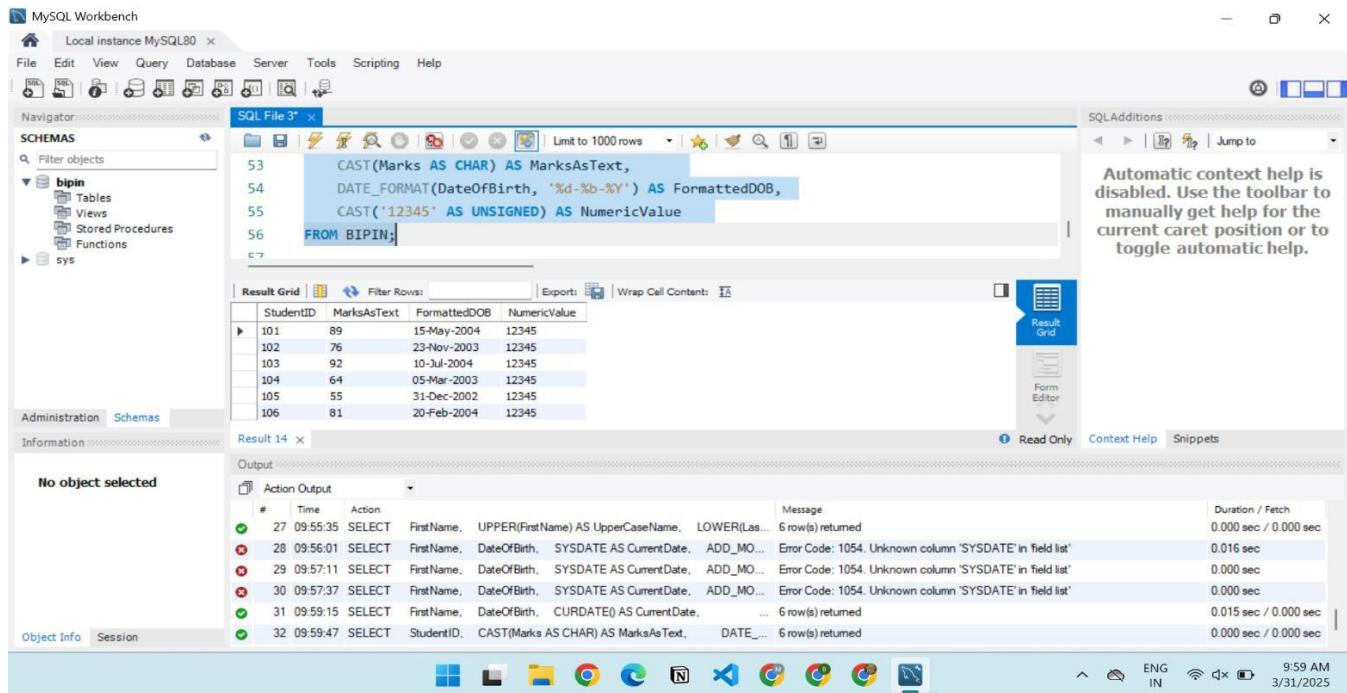
```

The results grid displays the following data:

FirstName	DateOfBirth	CurrentDate	SixMonthsLater	MonthsSinceBirth	StartOfYear
Bipin	2004-05-15	2025-03-31	2004-11-15	250	2025-01-01
Rahul	2003-11-23	2025-03-31	2004-05-23	256	2025-01-01
Sneha	2004-07-10	2025-03-31	2005-01-10	248	2025-01-01
Anjali	2003-03-05	2025-03-31	2003-09-05	264	2025-01-01
Karan	2002-12-31	2025-03-31	2003-06-30	267	2025-01-01
Priya	2004-02-20	2025-03-31	2004-08-20	253	2025-01-01

The status bar at the bottom right shows the date and time: 9:59 AM 3/31/2025.

CONVERSION FUNCUTION



The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator with the schema 'bipin' selected, showing Tables, Views, Stored Procedures, Functions, and sys. The main area contains a SQL editor tab titled 'SQL File 3*' with the following query:

```

53     CAST(Marks AS CHAR) AS MarksAsText,
54     DATE_FORMAT(DateOfBirth, '%d-%b-%Y') AS FormattedDOB,
55     CAST('12345' AS UNSIGNED) AS NumericValue
56 FROM BIPIN;
  
```

The result grid shows the following data:

StudentID	MarksAsText	FormattedDOB	NumericValue
101	89	15-May-2004	12345
102	76	23-Nov-2003	12345
103	92	10-Jul-2004	12345
104	64	05-Mar-2003	12345
105	55	31-Dec-2002	12345
106	81	20-Feb-2004	12345

The Output pane below shows the session history:

#	Time	Action	Message	Duration / Fetch
27	09:55:35	SELECT FirstName, UPPER(FirstName) AS UpperCaseName, LOWER(LastName) AS LowerCaseName FROM BIPIN;	6 row(s) returned	0.000 sec / 0.000 sec
28	09:56:01	SELECT FirstName, DateOfBirth, SYSDATE AS CurrentDate, ADD_MONTHS(DateOfBirth, 1) AS NextYear FROM BIPIN;	Error Code: 1054. Unknown column 'SYSDATE' in field list'	0.016 sec
29	09:57:11	SELECT FirstName, DateOfBirth, SYSDATE AS CurrentDate, ADD_MONTHS(DateOfBirth, 1) AS NextYear FROM BIPIN;	Error Code: 1054. Unknown column 'SYSDATE' in field list'	0.000 sec
30	09:57:37	SELECT FirstName, DateOfBirth, SYSDATE AS CurrentDate, ADD_MONTHS(DateOfBirth, 1) AS NextYear FROM BIPIN;	Error Code: 1054. Unknown column 'SYSDATE' in field list'	0.000 sec
31	09:59:15	SELECT FirstName, DateOfBirth, CURDATE() AS CurrentDate, ADD_MONTHS(DateOfBirth, 1) AS NextYear FROM BIPIN;	... 6 row(s) returned	0.015 sec / 0.000 sec
32	09:59:47	SELECT StudentID, CAST(Marks AS CHAR) AS MarksAsText, DATE_FORMAT(DateOfBirth, '%d-%b-%Y') AS FormattedDOB, CAST('12345' AS UNSIGNED) AS NumericValue FROM BIPIN;	6 row(s) returned	0.000 sec / 0.000 sec

EXPERIMENT 11

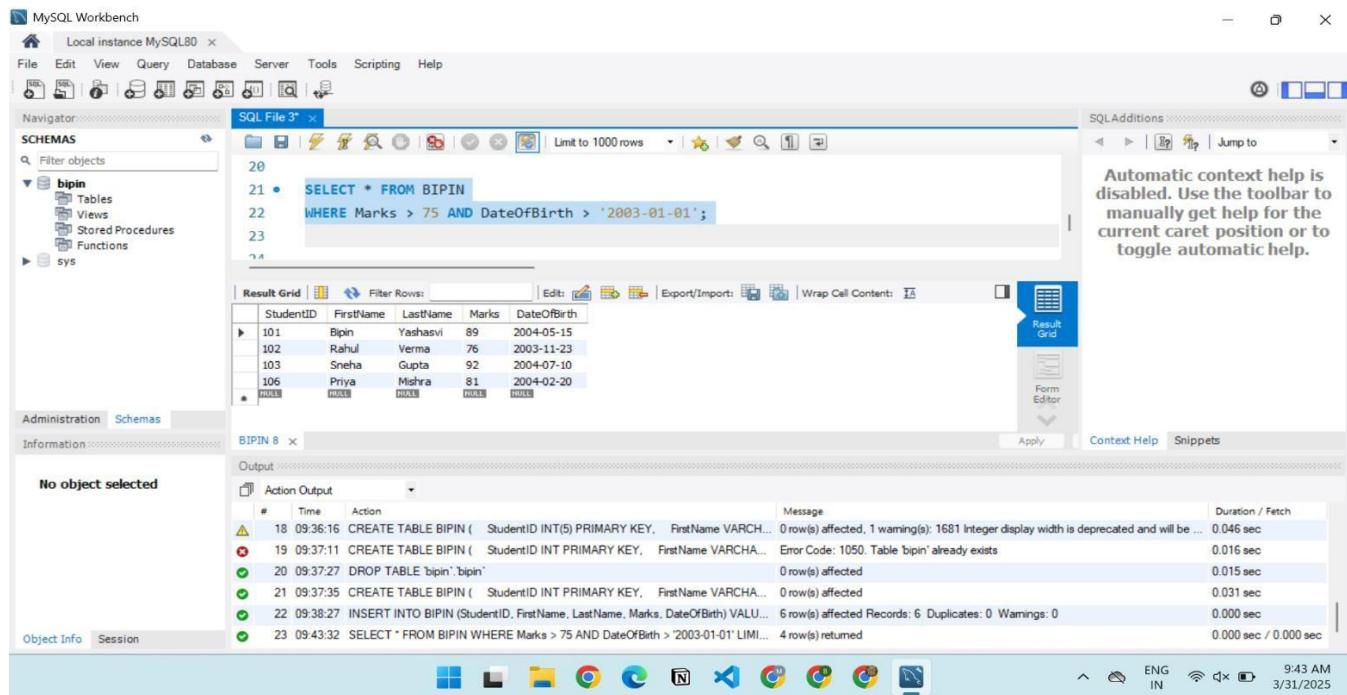
OBJECTIVE : Write a program to retrieve data using logical and arithmetic operators in SQL.

SOFTWARE USED : MySQL Workbench 8.0 CE

THEORY

SQL has many logical and arithmetic operators like AND, OR, NOT which can be used to retrieve specific data sets from a database.

AND OPERATOR



The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
SELECT * FROM BIPIN
WHERE Marks > 75 AND DateOfBirth > '2003-01-01';
```

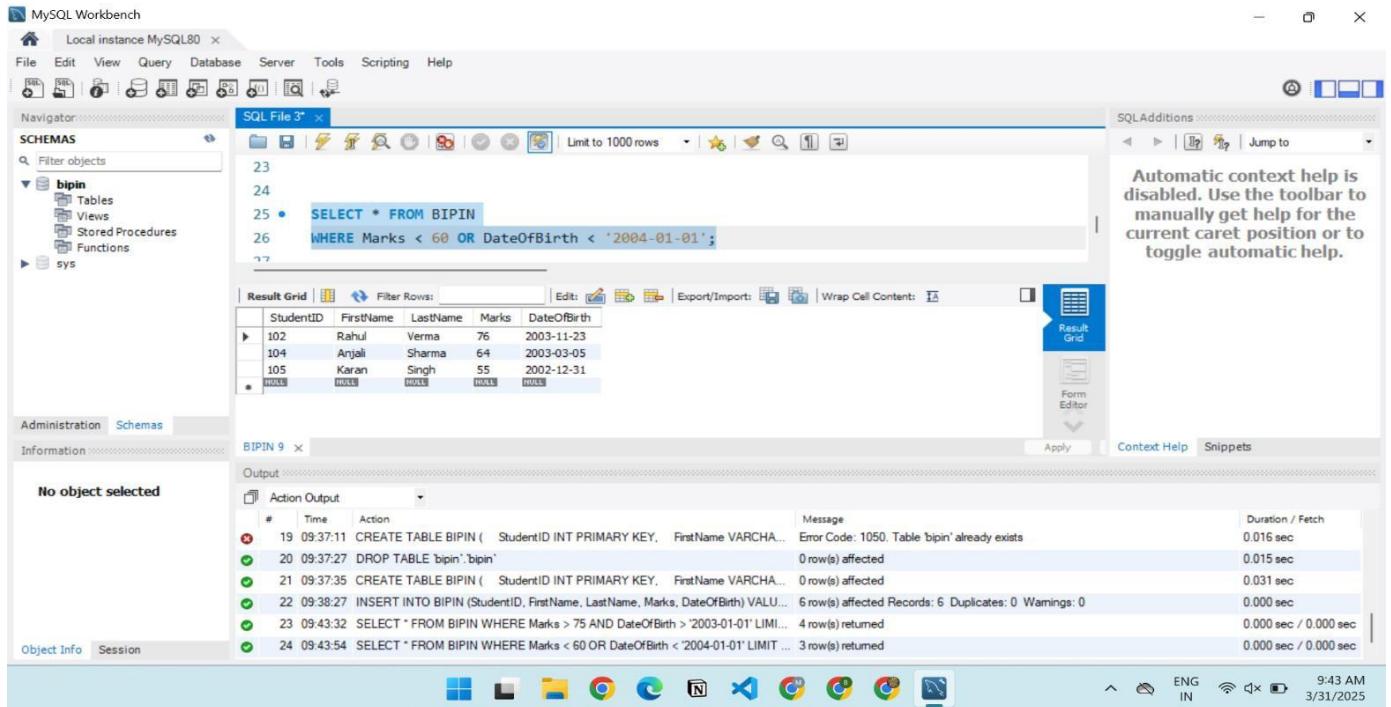
The result grid displays the following data:

StudentID	FirstName	LastName	Marks	DateOfBirth
101	Bipin	Yashavni	89	2004-05-15
102	Rahul	Verma	76	2003-11-23
103	Sneha	Gupta	92	2004-07-10
106	Priya	Mishra	81	2004-02-20
NULL	NULL	NULL	NULL	NULL

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
18	09:36:16	CREATE TABLE BIPIN (StudentID INT(5) PRIMARY KEY, FirstName VARCHAR...) ENGINE=InnoDB	0 row(s) affected, 1 warning(s); 1681 Integer display width is deprecated and will be ...	0.046 sec
19	09:37:11	CREATE TABLE BIPIN (StudentID INT PRIMARY KEY, FirstName VARCHAR...) ENGINE=InnoDB	Error Code: 1050. Table 'bipin' already exists	0.016 sec
20	09:37:27	DROP TABLE `bipin`	0 row(s) affected	0.015 sec
21	09:37:35	CREATE TABLE BIPIN (StudentID INT PRIMARY KEY, FirstName VARCHAR...) ENGINE=InnoDB	0 row(s) affected	0.031 sec
22	09:38:27	INSERT INTO BIPIN (StudentID, FirstName, LastName, Marks, DateOfBirth) VALUES (101, 'Bipin', 'Yashavni', 89, '2004-05-15')	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec / 0.000 sec
23	09:43:32	SELECT * FROM BIPIN WHERE Marks > 75 AND DateOfBirth > '2003-01-01' LIMIT 100	4 row(s) returned	0.000 sec / 0.000 sec

OR OPERATOR



MySQL Workbench - Local Instance MySQL80

SQL File 3*

```

23
24
25 •  SELECT * FROM BIPIN
26 WHERE Marks < 60 OR DateOfBirth < '2004-01-01';
27
  
```

Result Grid

StudentID	FirstName	LastName	Marks	DateOfBirth
102	Rahul	Verma	76	2003-11-23
104	Anjali	Sharma	64	2003-03-05
105	Karan	Singh	55	2002-12-31
*	HULL	HULL	HULL	HULL

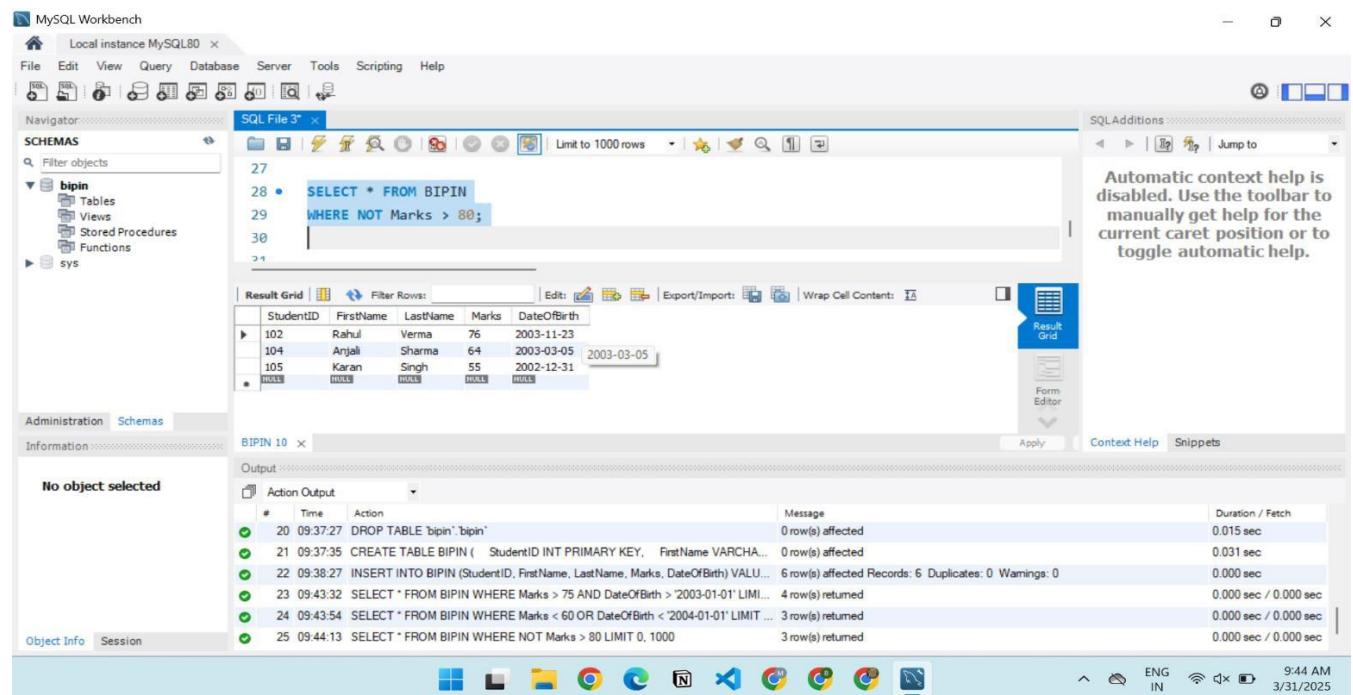
BIPIN 9 ×

Output

#	Time	Action	Message	Duration / Fetch
19	09:37:11	CREATE TABLE BIPIN (StudentID INT PRIMARY KEY, FirstName VARCHAR...	Error Code: 1050. Table 'bipin' already exists	0.016 sec
20	09:37:27	DROP TABLE bipin.bipin'	0 row(s) affected	0.015 sec
21	09:37:35	CREATE TABLE BIPIN (StudentID INT PRIMARY KEY, FirstName VARCHAR...	0 row(s) affected	0.031 sec
22	09:38:27	INSERT INTO BIPIN (StudentID, FirstName, LastName, Marks, DateOfBirth) VALUES (102, 'Rahul', 'Verma', 76, '2003-11-23')	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec
23	09:43:32	SELECT * FROM BIPIN WHERE Marks > 75 AND DateOfBirth > '2003-01-01' LIMIT 1	4 row(s) returned	0.000 sec / 0.000 sec
24	09:43:54	SELECT * FROM BIPIN WHERE Marks < 60 OR DateOfBirth < '2004-01-01' LIMIT 1	3 row(s) returned	0.000 sec / 0.000 sec

9:43 AM 3/31/2025

NOT OPERATOR



MySQL Workbench - Local Instance MySQL80

SQL File 3*

```

27
28 •  SELECT * FROM BIPIN
29 WHERE NOT Marks > 80;
30
  
```

Result Grid

StudentID	FirstName	LastName	Marks	DateOfBirth
102	Rahul	Verma	76	2003-11-23
104	Anjali	Sharma	64	2003-03-05
105	Karan	Singh	55	2002-12-31
*	HULL	HULL	HULL	HULL

BIPIN 10 ×

Output

#	Time	Action	Message	Duration / Fetch
20	09:37:27	DROP TABLE bipin.bipin'	0 row(s) affected	0.015 sec
21	09:37:35	CREATE TABLE BIPIN (StudentID INT PRIMARY KEY, FirstName VARCHAR...	0 row(s) affected	0.031 sec
22	09:38:27	INSERT INTO BIPIN (StudentID, FirstName, LastName, Marks, DateOfBirth) VALUES (102, 'Rahul', 'Verma', 76, '2003-11-23')	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec
23	09:43:32	SELECT * FROM BIPIN WHERE Marks > 75 AND DateOfBirth > '2003-01-01' LIMIT 1	4 row(s) returned	0.000 sec / 0.000 sec
24	09:43:54	SELECT * FROM BIPIN WHERE Marks < 60 OR DateOfBirth < '2004-01-01' LIMIT 1	3 row(s) returned	0.000 sec / 0.000 sec
25	09:44:13	SELECT * FROM BIPIN WHERE NOT Marks > 80 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

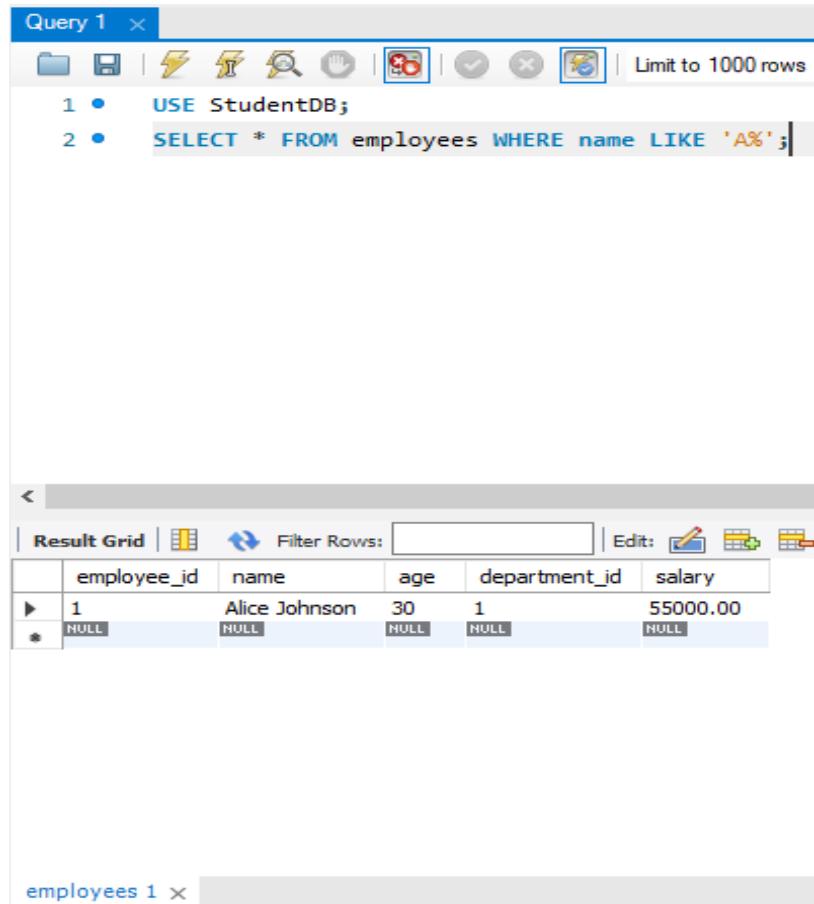
9:44 AM 3/31/2025

EXPERIMENT-14

Q: To implement Different operators(LIKE, BETWEEN, IN, WILDCARDS)

LIKE Operator:

Find employees whose names start with the letter "A":



The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, the query editor window displays two lines of SQL code:

```
1 • USE StudentDB;  
2 • SELECT * FROM employees WHERE name LIKE 'A%';
```

Below the query editor is the result grid window. It has a header row with columns: employee_id, name, age, department_id, and salary. The data grid contains one visible row for Alice Johnson, with the following values:

	employee_id	name	age	department_id	salary
▶	1	Alice Johnson	30	1	55000.00
*	NULL	NULL	NULL	NULL	NULL

BETWEEN Operator:

Find employees whose salary is between \$60,000 and \$80,000:

The screenshot shows the MySQL Workbench interface with two tabs: "Query 1" and "employees 2". The "Query 1" tab contains the following SQL code:

```

Query 1 ×
1 • USE StudentDB;
2 • SELECT * FROM employees WHERE salary BETWEEN 60000 AND 80000;
3

```

The "Result Grid" under "employees 2" displays the following data:

	employee_id	name	age	department_id	salary
▶	2	Bob Smith	40	2	60000.00
3		Charlie Davis	35	3	75000.00
4		David Wilson	45	4	65000.00
5		Eva Thomas	28	5	72000.00
6		Frank Miller	50	2	80000.00
8		Hannah White	25	3	68000.00
*	NULL	NULL	NULL	NULL	NULL

IN Operator

Find employees who work in the "HR", "Sales", or "IT" departments:

The screenshot shows the MySQL Workbench interface with three tabs: "Query 1", "employees 2", and "employees 3". The "Query 1" tab contains the following SQL code:

```

Query 1 ×
1 • USE StudentDB;
2 • SELECT * FROM employees WHERE department_id IN (1, 2, 3);
3

```

The "Result Grid" under "employees 3" displays the following data:

	employee_id	name	age	department_id	salary
▶	1	Alice Johnson	30	1	55000.00
7		Grace Lee	38	1	48000.00
2		Bob Smith	40	2	60000.00
6		Frank Miller	50	2	80000.00
3		Charlie Davis	35	3	75000.00
8		Hannah White	25	3	68000.00
*	NULL	NULL	NULL	NULL	NULL

Wildcards:

Find employees whose names start with "D" and have any characters after it (wildcard %):

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, the 'Query 1' tab is active, displaying two SQL statements:

```
1 • USE StudentDB;
2 • SELECT * FROM employees WHERE name LIKE 'D%';
```

Below the query editor is a 'Result Grid' window. It has a header row with columns: employee_id, name, age, department_id, and salary. The data row shows:

	employee_id	name	age	department_id	salary
▶	4	David Wilson	45	4	65000.00
*	NUL	NUL	NUL	NUL	NUL

At the bottom of the interface, there is another tab labeled 'employees 5'.

16. Introduction of VIEW. How to create view. Explain with example.

In the context of databases, a view is a virtual table that consists of a subset of data from one or more tables. Views are useful for various purposes, such as simplifying complex queries, providing a layer of security by restricting access to certain columns or rows, and presenting data in a particular format for reporting or analysis.

Creating a view in a typical MySQL database -

```
```CREATE VIEW view_name AS
```

```
SELECT column1, column2,
```

```
FROM table_name
```

```
WHERE condition;```
```

```
mysql> select * from new;
+---+-----+---+
| id | name | age |
+---+-----+---+
| 1 | akash_rattan | 56 |
| 2 | oggy | 30 |
| 3 | jack | 10 |
| 4 | panther | 5 |
+---+-----+---+
4 rows in set (0.00 sec)

mysql> create view second as select name,age from new where age>30;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from second;
+-----+---+
| name | age |
+-----+---+
| akash_rattan | 56 |
+-----+---+
1 row in set (0.00 sec)
```

## Exercise-1

### Sales Table:

```
CREATE TABLE Sales (
 sale_id INT PRIMARY KEY,
 product_id INT,
 quantity_sold INT,
 sale_date DATE,
 total_price DECIMAL(10, 2)
);
```

### Products Table:

```
CREATE TABLE Products (
 product_id INT PRIMARY KEY,
 product_name VARCHAR(100),
 category VARCHAR(50),
 unit_price DECIMAL(10, 2)
);
```

### Questions:

1. Retrieve all columns from the Sales table.
2. Retrieve the product name and unit\_price from the Products table.
3. Filter the Sales table to show only sales with a total price greater than \$100.
4. Filter the Products table to show only products in the 'Electronics' category.
5. Retrieve the sale\_id and total\_price from the Sales table for sales made on January 3, 2024.
6. Retrieve the product\_id and product name from the Products table for products with a unit price greater than \$100.

## 1. Retrieve all columns from the Sales table.

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

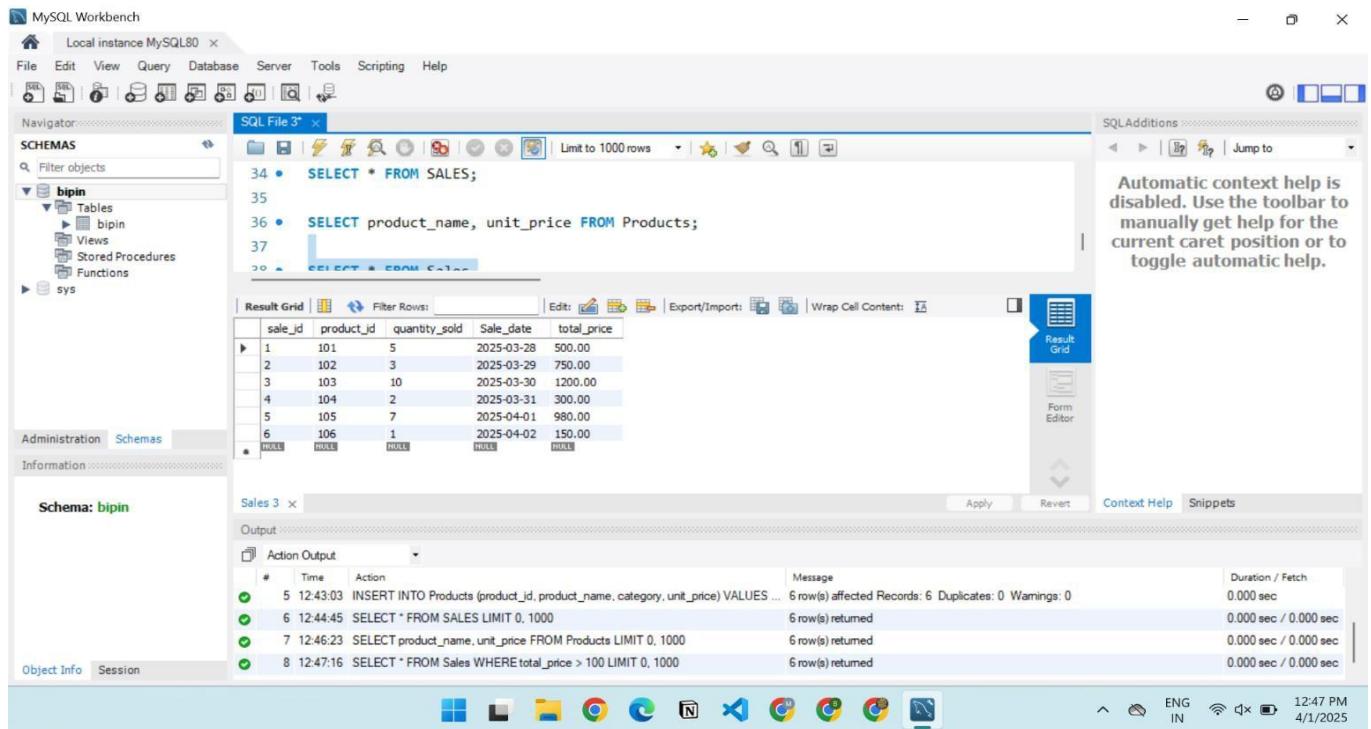
- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (selected), Filter objects, bipin (Tables, Views, Stored Procedures, Functions).
- SQL Editor:** SQL File 3\*, Line 34: `SELECT * FROM SALES;`
- Result Grid:** Displays the results of the query, showing 6 rows of data with columns: sale\_id, product\_id, quantity\_sold, Sale\_date, total\_price.
- Action Output:** Shows the history of actions taken, including the creation of the Products table and the insertion of data into the Sales and Products tables.
- System Status:** Shows the date and time as 12:44 PM on 4/1/2025.

## 2. Retrieve the product\_name and unit\_price from the Products table.

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (selected), Filter objects, bipin (Tables, Views, Stored Procedures, Functions).
- SQL Editor:** SQL File 3\*, Lines 34-36:
  - Line 34: `SELECT * FROM SALES;`
  - Line 35: `SELECT * FROM SALES LIMIT 0, 1000;`
  - Line 36: `SELECT product_name, unit_price FROM Products;`
- Result Grid:** Displays the results of the query in Line 36, showing 6 rows of data with columns: product\_name, unit\_price.
- Action Output:** Shows the history of actions taken, including the creation of the Products table and the insertion of data into the Sales and Products tables.
- System Status:** Shows the date and time as 12:46 PM on 4/1/2025.

### 3. Filter the Sales table to show only sales with a total price greater than \$100.



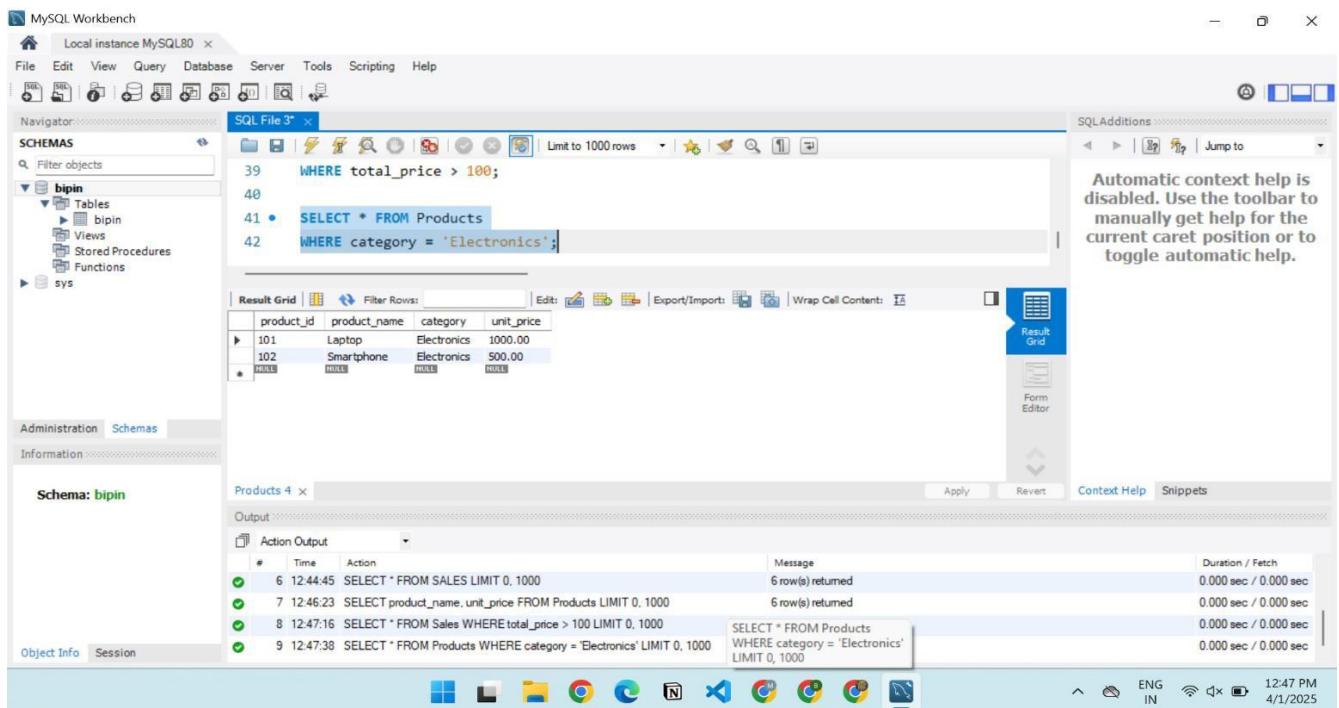
The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **bipin** with tables **bipin**, **Views**, **Stored Procedures**, and **Functions**.
- SQL Editor:** Contains the following SQL code:
 

```
34 • SELECT * FROM SALES;
35
36 • SELECT product_name, unit_price FROM Products;
37
38 • SELECT * FROM Sales WHERE total_price > 100;
```
- Result Grid:** Displays the results of the final query, showing 6 rows of sales data where the total price is greater than 100. The columns are **sale\_id**, **product\_id**, **quantity\_sold**, **Sale\_date**, and **total\_price**. The data includes rows for products 101 through 106.
- Output:** Shows the execution history with the following log entries:
 

#	Time	Action	Message	Duration / Fetch
5	12:43:03	INSERT INTO Products (product_id, product_name, category, unit_price) VALUES ...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec
6	12:44:45	SELECT * FROM SALES LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
7	12:46:23	SELECT product_name, unit_price FROM Products LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
8	12:47:16	SELECT * FROM Sales WHERE total_price > 100 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

### 4. Filter the Products table to show only products in the 'Electronics' category.



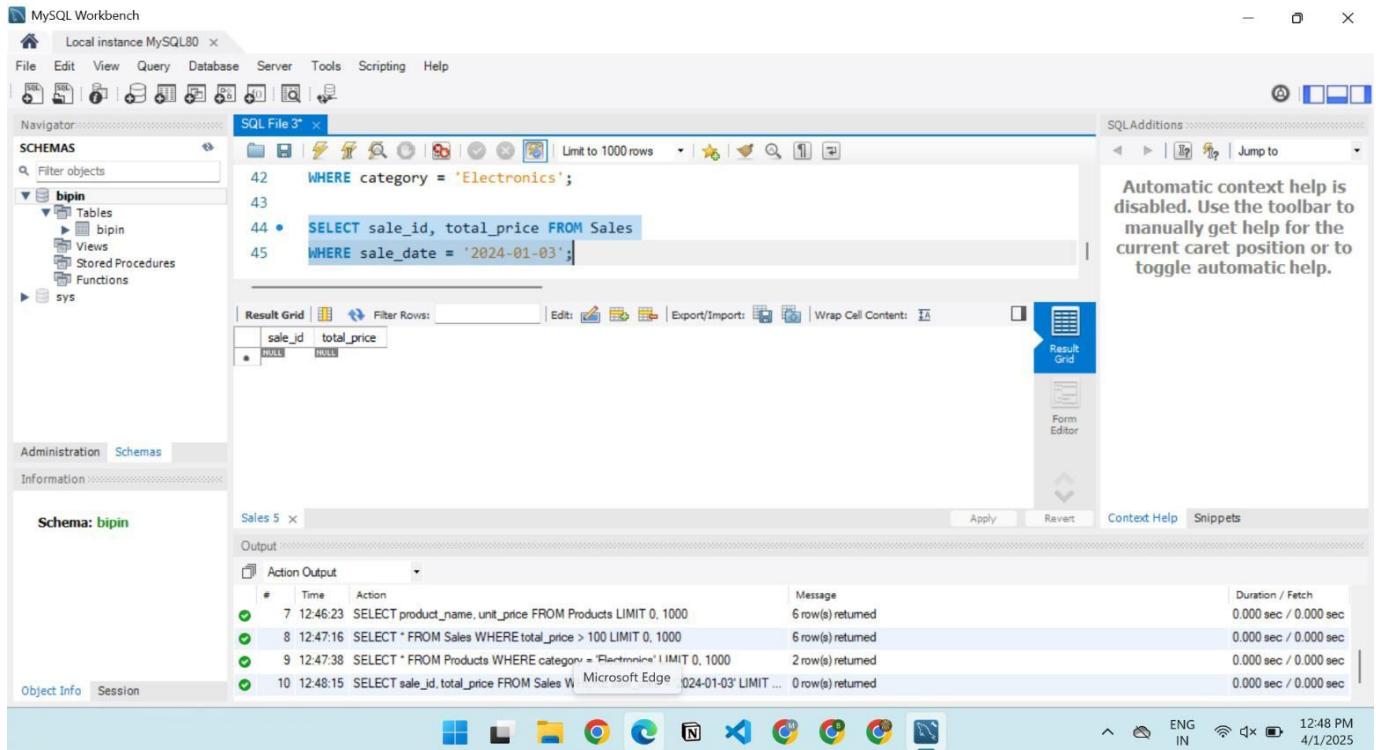
The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **bipin** with tables **bipin**, **Views**, **Stored Procedures**, and **Functions**.
- SQL Editor:** Contains the following SQL code:
 

```
39 WHERE total_price > 100;
40
41 • SELECT * FROM Products
42 WHERE category = 'Electronics';
```
- Result Grid:** Displays the results of the query, showing 2 rows of products in the 'Electronics' category. The columns are **product\_id**, **product\_name**, **category**, and **unit\_price**. The data includes rows for products 101 and 102.
- Output:** Shows the execution history with the following log entries:
 

#	Time	Action	Message	Duration / Fetch
6	12:44:45	SELECT * FROM SALES LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
7	12:46:23	SELECT product_name, unit_price FROM Products LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
8	12:47:16	SELECT * FROM Sales WHERE total_price > 100 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
9	12:47:38	SELECT * FROM Products WHERE category = 'Electronics' LIMIT 0, 1000	SELECT * FROM Products WHERE category = 'Electronics' LIMIT 0, 1000	0.000 sec / 0.000 sec

**5. Retrieve the sale\_id and total\_price from the Sales table for sales made on January 3, 2024.**



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

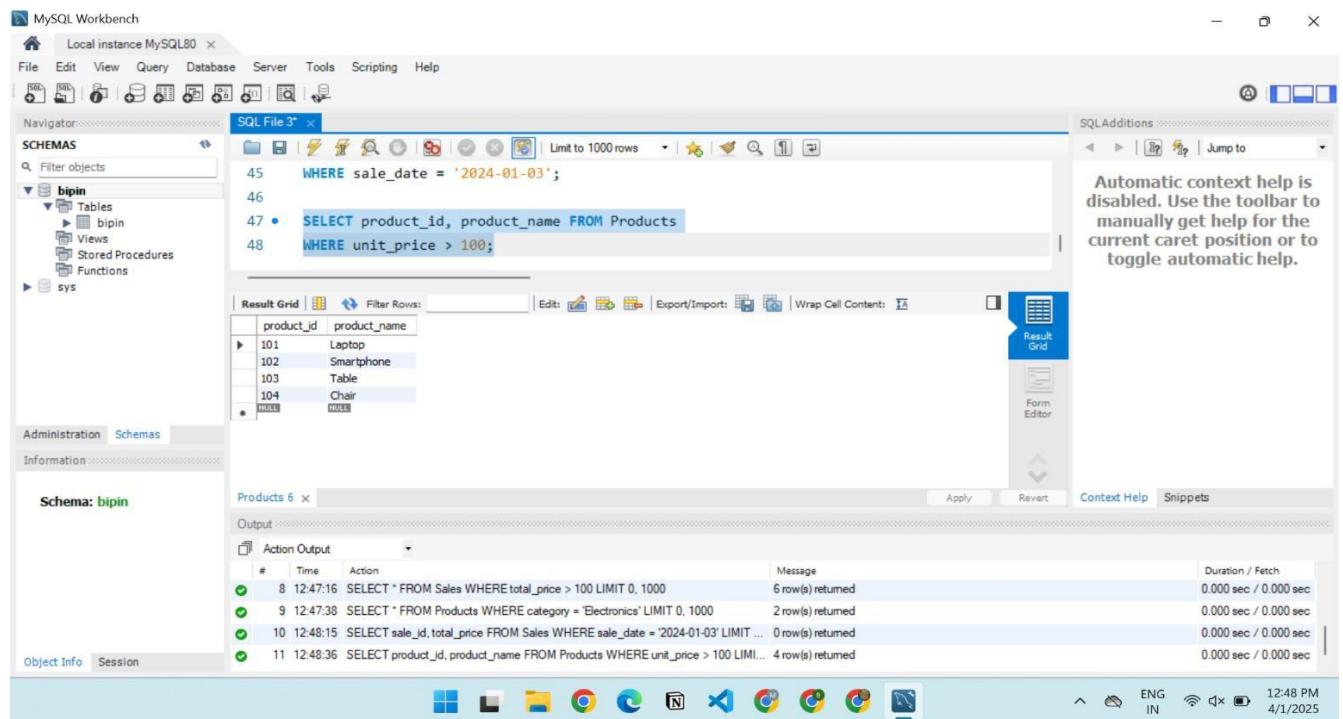
- SQL Editor:** Contains the following SQL code:
 

```
42 WHERE category = 'Electronics';
43
44 • SELECT sale_id, total_price FROM Sales
45 WHERE sale_date = '2024-01-03';
```
- Result Grid:** Displays the results of the query:
 

sale_id	total_price
NULL	NULL
- Output Panel:** Shows the action log with the following entries:
 

#	Time	Action	Message	Duration / Fetch
7	12:46:23	SELECT product_name, unit_price FROM Products LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
8	12:47:16	SELECT * FROM Sales WHERE total_price > 100 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
9	12:47:38	SELECT * FROM Products WHERE category = 'Electronics' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
10	12:48:15	SELECT sale_id, total_price FROM Sales WHERE sale_date = '2024-01-03' LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

**6. Retrieve the product\_id and product\_name from the Products table for products with a unit price greater than \$100.**



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

- SQL Editor:** Contains the following SQL code:
 

```
45 WHERE sale_date = '2024-01-03';
46
47 • SELECT product_id, product_name FROM Products
48 WHERE unit_price > 100;
```
- Result Grid:** Displays the results of the query:
 

product_id	product_name
101	Laptop
102	Smartphone
103	Table
104	Chair
105	Monitor
- Output Panel:** Shows the action log with the following entries:
 

#	Time	Action	Message	Duration / Fetch
8	12:47:16	SELECT * FROM Sales WHERE total_price > 100 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
9	12:47:38	SELECT * FROM Products WHERE category = 'Electronics' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
10	12:48:15	SELECT sale_id, total_price FROM Sales WHERE sale_date = '2024-01-03' LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
11	12:48:36	SELECT product_id, product_name FROM Products WHERE unit_price > 100 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

## SQL Exercise-2

**EmployeeInfo Table:**

EmpID	EmpFname	EmpLname	Department	Project	Address	DOB	Gender
1	Sanjay	Mehra	HR	P1	Hyderabad(HYD)	01/12/1976	M
2	Ananya	Mishra	Admin	P2	Delhi(DEL)	02/05/1968	F
3	Rohan	Diwan	Account	P3	Mumbai(BOM)	01/01/1980	M
4	Sonia	Kulkarni	HR	P1	Hyderabad(HYD)	02/05/1992	F
5	Ankit	Kapoor	Admin	P2	Delhi(DEL)	03/07/1994	M

**EmployeePosition Table:**

EmpID	EmpPosition	DateOfJoining	Salary
1	Manager	01/05/2024	500000
2	Executive	02/05/2024	75000
3	Manager	01/05/2024	90000
2	Lead	02/05/2024	85000
1	Executive	01/05/2024	300000

**Consider the above-mentioned tables, write the answer of following SQL queries:**

1. Write a query to fetch the EmpFname from the EmployeeInfo table in the upper case and use the ALIAS name as EmpName.
2. Write a query to fetch the number of employees working in the department ‘HR’.
3. Write a query to get the current date.
4. Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.
5. Write a query to fetch only the place name (string before brackets) from the Address column of EmployeeInfo table.
6. Write a query to create a new table that consists of data and structure copied from the other table.
7. Write q query to find all the employees whose salary is between 50000 to 100000.
8. Write a query to find the names of employees that begin with ‘S’
9. Write a query to fetch top N records.
10. Write a query to retrieve the EmpFname and EmpLname in a single column as “FullName”.  
The first name and the last name must be separated with space.

## Exercise-2

### **EmployeeInfo Table:**

```
CREATE TABLE EmployeeInfo (
 EmpID INT PRIMARY KEY,
 EmpFname VARCHAR(50),
 EmpLname VARCHAR(50),
 Department VARCHAR(50),
 Project VARCHAR(50),
 Address VARCHAR(100),
 DOB DATE,
 Gender VARCHAR(10)
);
```

### **EmployeePosition Table:**

```
CREATE TABLE Employeeposition (
 ID INT AUTO_INCREMENT PRIMARY KEY,
 EmpID INT,
 EmpPosition VARCHAR(50),
 DateOfJoining DATE,
 Salary DECIMAL(10, 2)
);
```

## Questions:

1. Write a query to fetch the EmpFname from the EmployeeInfo table in the upper case and use the ALIAS name as EmpName.

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following query is run:

```
39
40 • SELECT UPPER(EmpFname) AS EmpName
41 FROM EmployeeInfo;
42
```

The Result Grid shows the output:

EmpName
SANJAY
ANANYA
ROHAN
SONIA
ANKIT

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
8	00:09:44	CREATE TABLE Employeeposition ( ID INT AUTO_INCREMENT PRIMARY KEY...	Error Code: 1050. Table 'employeeposition' already exists	0.00 sec
9	00:10:04	DROP TABLE 'bipin'.`employeeposition`	0 row(s) affected	0.031 sec
10	00:10:10	DROP TABLE bipin.`employeeposition`	Error Code: 1051. Unknown table 'bipin.employeeposition'	0.00 sec
11	00:10:23	CREATE TABLE Employeeposition ( ID INT AUTO_INCREMENT PRIMARY KEY...	0 row(s) affected	0.016 sec
12	00:10:29	INSERT INTO EmployeePosition (EmpID, EmpPosition, DateOfJoining, Salary) VAL...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
13	00:11:29	SELECT UPPER(EmpFname) AS EmpName FROM EmployeeInfo LIMIT 0, 1000	5 row(s) returned	0.00 sec / 0.000 sec
14	00:12:31	SELECT COUNT(*) AS HR_EmployeeCount FROM EmployeeInfo WHERE Depart...	1 row(s) returned	0.00 sec / 0.000 sec

2. Write a query to fetch the number of employees working in the department 'HR'.

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following query is run:

```
42
43 • SELECT COUNT(*) AS HR_EmployeeCount
44 FROM EmployeeInfo
45 WHERE Department = 'HR';
```

The Result Grid shows the output:

HR_EmployeeCount
2

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
9	00:10:04	DROP TABLE 'bipin'.`employeeposition`	0 row(s) affected	0.031 sec
10	00:10:10	DROP TABLE bipin.`employeeposition`	Error Code: 1051. Unknown table 'bipin.employeeposition'	0.00 sec
11	00:10:23	CREATE TABLE Employeeposition ( ID INT AUTO_INCREMENT PRIMARY KEY...	0 row(s) affected	0.016 sec
12	00:10:29	INSERT INTO EmployeePosition (EmpID, EmpPosition, DateOfJoining, Salary) VAL...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
13	00:11:29	SELECT UPPER(EmpFname) AS EmpName FROM EmployeeInfo LIMIT 0, 1000	5 row(s) returned	0.00 sec / 0.000 sec
14	00:12:31	SELECT COUNT(*) AS HR_EmployeeCount FROM EmployeeInfo WHERE Depart...	1 row(s) returned	0.00 sec / 0.000 sec

2023305225

### 3. Write a query to get the current date.

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, a query is being run:

```
44 FROM EmployeeInfo
45 WHERE Department = 'HR';
46
47 • SELECT CURDATE() AS CurrentDate;
```

The Result Grid shows the output:

CurrentDate
2025-04-15

In the Output pane, the following log entries are visible:

#	Time	Action	Message	Duration / Fetch
10	00:10:10	DROP TABLE `bipin`.`employeeposition`	Error Code: 1051. Unknown table `bipin.employeeposition`	0.000 sec
11	00:10:23	CREATE TABLE EmployeePosition ( ID INT AUTO_INCREMENT PRIMARY KEY, ... )	0 row(s) affected	0.016 sec
12	00:10:29	INSERT INTO EmployeePosition (EmpID, EmpPosition, DateOfJoining, Salary) VALUES (1, 'Manager', '2023-04-15', 5000)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
13	00:11:29	SELECT UPPER(EmpName) AS EmpName FROM EmployeeInfo LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
14	00:12:31	SELECT COUNT(*) AS HR_EmployeeCount FROM EmployeeInfo WHERE Department = 'HR'	1 row(s) returned	0.000 sec / 0.000 sec
15	00:13:03	SELECT CURDATE() AS CurrentDate LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

### 4. Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, a query is being run:

```
47 • SELECT CURDATE() AS CurrentDate;
48
49 • SELECT SUBSTRING(EmpLname, 1, 4) AS FirstFourChars
50 FROM EmployeeInfo;
```

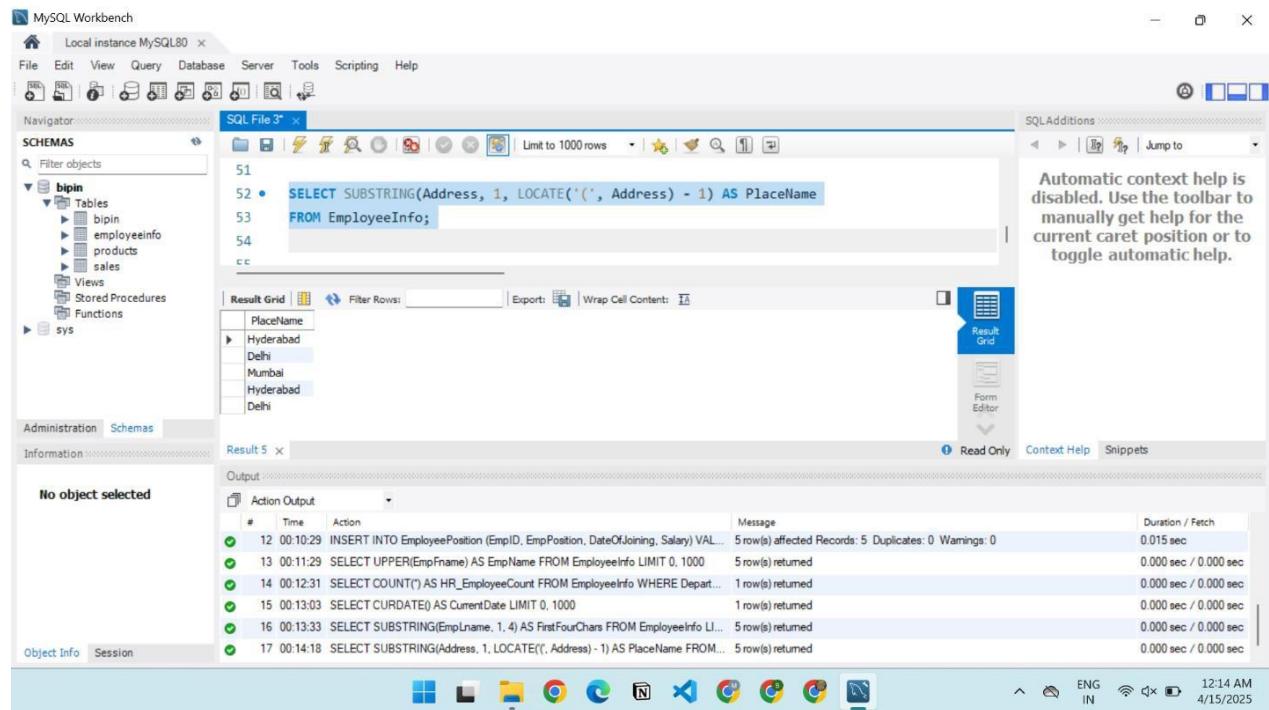
The Result Grid shows the output:

FirstFourChars
Mehr
Mish
Divya
Kulk
Kapo

In the Output pane, the following log entries are visible:

#	Time	Action	Message	Duration / Fetch
11	00:10:23	CREATE TABLE EmployeePosition ( ID INT AUTO_INCREMENT PRIMARY KEY, ... )	0 row(s) affected	0.016 sec
12	00:10:29	INSERT INTO EmployeePosition (EmpID, EmpPosition, DateOfJoining, Salary) VALUES (1, 'Manager', '2023-04-15', 5000)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
13	00:11:29	SELECT UPPER(EmpName) AS EmpName FROM EmployeeInfo LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
14	00:12:31	SELECT COUNT(*) AS HR_EmployeeCount FROM EmployeeInfo WHERE Department = 'HR'	1 row(s) returned	0.000 sec / 0.000 sec
15	00:13:03	SELECT CURDATE() AS CurrentDate LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	00:13:33	SELECT SUBSTRING(EmpLname, 1, 4) AS FirstFourChars FROM EmployeeInfo LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

**5. Write a query to fetch only the place name (string before brackets) from the Address column of EmployeeInfo table.**

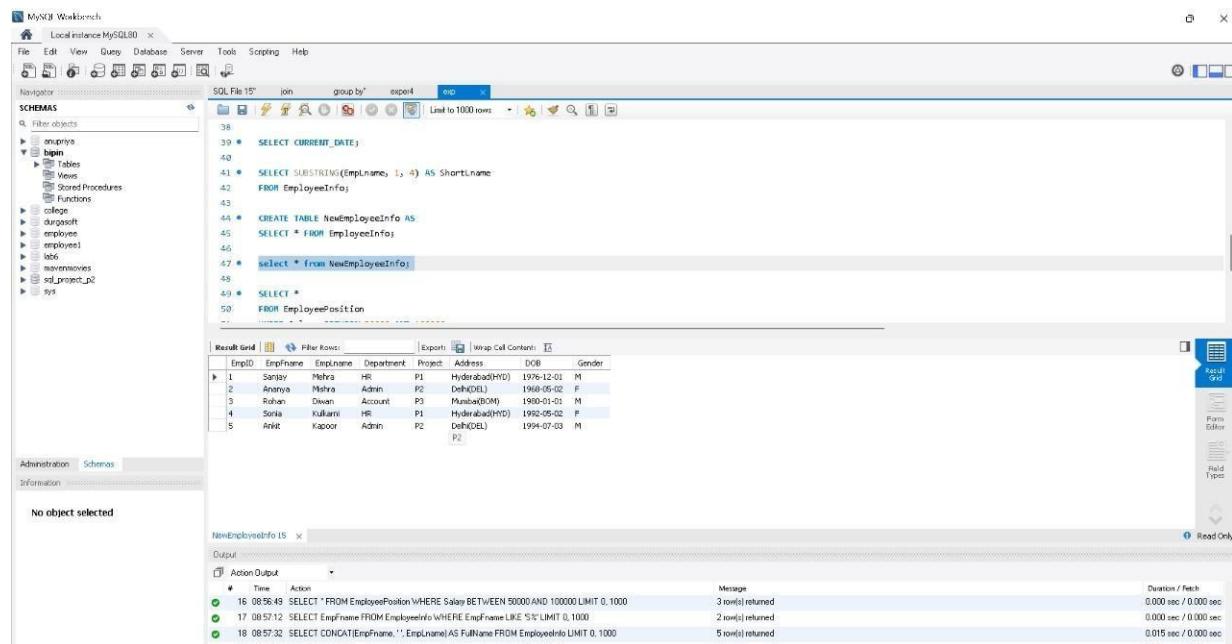


The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (selected), showing the bipin schema with tables like bipin, employeeinfo, products, sales, etc.
- SQL Editor:** SQL File 3\*, containing the query:
 

```
51 • SELECT SUBSTRING(Address, 1, LOCATE('(', Address) - 1) AS PlaceName
52 FROM EmployeeInfo;
```
- Result Grid:** Shows the output of the query, displaying rows for Hyderabad, Delhi, Mumbai, Hyderabad, and Delhi.
- Output Window:** Action Output pane showing the execution log for the session.
- System Bar:** Includes icons for file operations, browser, and system status (ENG IN, 12:14 AM, 4/15/2025).

**6. Write a query to create a new table that consists of data and structure copied from the other table.**



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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (selected), showing the bipin schema with tables like employees, EmployeeInfo, products, sales, etc.
- SQL Editor:** SQL File 15\*, containing the creation script for NewEmployeeInfo:
 

```
36 • SELECT CURRENT_DATE;
37
38 • CREATE TABLE NewEmployeeInfo AS
39 SELECT * FROM EmployeeInfo;
40
41 • select * from NewEmployeeInfo;
42
43
44 • SELECT *
45 FROM EmployeePosition;
```
- Result Grid:** Shows the data copied from EmployeeInfo into NewEmployeeInfo, including columns EmpID, EmpName, Employee, Department, Project, Address, DOB, and Gender.
- Action Output:** Shows the execution log for the session, including the creation of the table and the insertion of data.

## 7. Write q query to find all the employees whose salary is between 50000 to 100000.

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following query is run:

```
SELECT *
FROM EmployeePosition
WHERE Salary BETWEEN 50000 AND 100000;
```

The Result Grid displays the following data:

ID	EmpID	EmpPosition	DateOfJoining	Salary
2	2	Executive	2024-05-02	75000.00
3	3	Manager	2024-05-01	90000.00
4	2	Lead	2024-05-02	85000.00
*	HULL	HULL	HULL	HULL

The Output tab shows the following log entries:

#	Time	Action	Message	Duration / Fetch
16	00:13:33	SELECT SUBSTRING(EmpLname, 1, 4) AS FirstFourChars FROM EmployeeInfo U...	5 row(s) returned	0.000 sec / 0.000 sec
17	00:14:18	SELECT SUBSTRING(Address, 1, LOCATE('(', Address) - 1) AS PlaceName FROM...	5 row(s) returned	0.000 sec / 0.000 sec
18	00:14:42	CREATE TABLE NewTable AS SELECT * FROM ExistingTable	Error Code: 1146. Table 'bipin.existingtable' doesn't exist	0.000 sec
19	00:15:26	SELECT * INTO NewTable FROM ExistingTable	Error Code: 1327. Undeclared variable: NewTable	0.000 sec
20	00:15:53	SELECT * INTO NewTable FROM Employeeposition	Error Code: 1327. Undeclared variable: NewTable	0.000 sec
21	00:17:28	SELECT * FROM EmployeePosition WHERE Salary BETWEEN 50000 AND 10000...	3 row(s) returned	0.000 sec / 0.000 sec

## 8. Write a query to find the names of employees that begin with 'S'.

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following query is run:

```
SELECT EmpFname
FROM EmployeeInfo
WHERE EmpFname LIKE 'S%';
```

The Result Grid displays the following data:

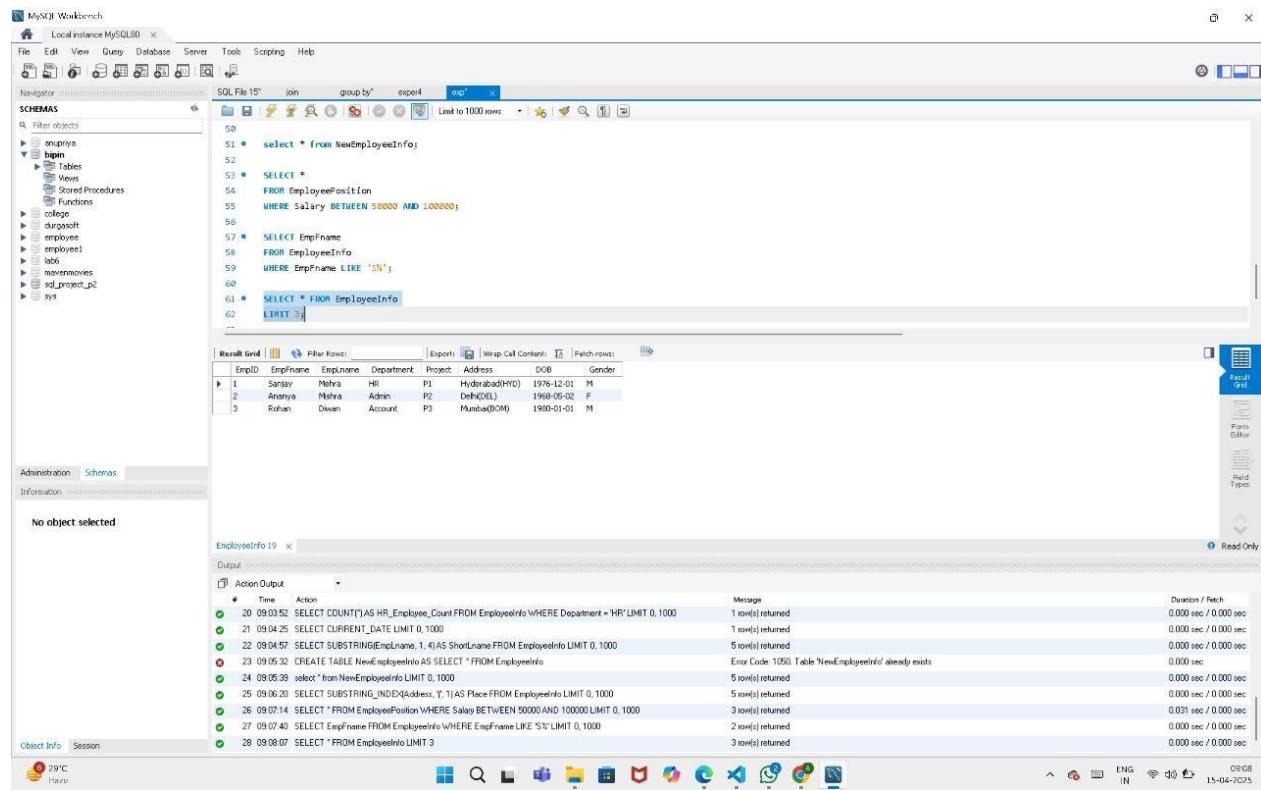
EmpFname
Sanjay
Sonia

The Output tab shows the following log entries:

#	Time	Action	Message	Duration / Fetch
17	00:14:18	SELECT SUBSTRING(Address, 1, LOCATE('(', Address) - 1) AS PlaceName FROM...	5 row(s) returned	0.000 sec / 0.000 sec
18	00:14:42	CREATE TABLE NewTable AS SELECT * FROM ExistingTable	Error Code: 1146. Table 'bipin.existingtable' doesn't exist	0.000 sec
19	00:15:26	SELECT * INTO NewTable FROM ExistingTable	Error Code: 1327. Undeclared variable: NewTable	0.000 sec
20	00:15:53	SELECT * INTO NewTable FROM Employeeposition	Error Code: 1327. Undeclared variable: NewTable	0.000 sec
21	00:17:28	SELECT * FROM EmployeePosition WHERE Salary BETWEEN 50000 AND 10000...	3 row(s) returned	0.000 sec / 0.000 sec
22	00:17:59	SELECT EmpFname FROM EmployeeInfo WHERE EmpFname LIKE 'S%' LIMIT 0, ...	2 row(s) returned	0.016 sec / 0.000 sec

2023305225

## 9. Write a query to fetch top N records.



The screenshot shows the MySQL Workbench interface with a SQL editor containing the following query:

```

50
51 • select * from NewEmployeeInfo;
52
53 • SELECT *
54 FROM EmployeePosition
55 WHERE Salary BETWEEN $0000 AND 10000;
56
57 • SELECT EmpName
58 FROM EmployeeInfo
59 WHERE EmpName LIKE '%N%';
60
61 • SELECT * FROM EmployeeInfo
62 LIMIT 3;
--
```

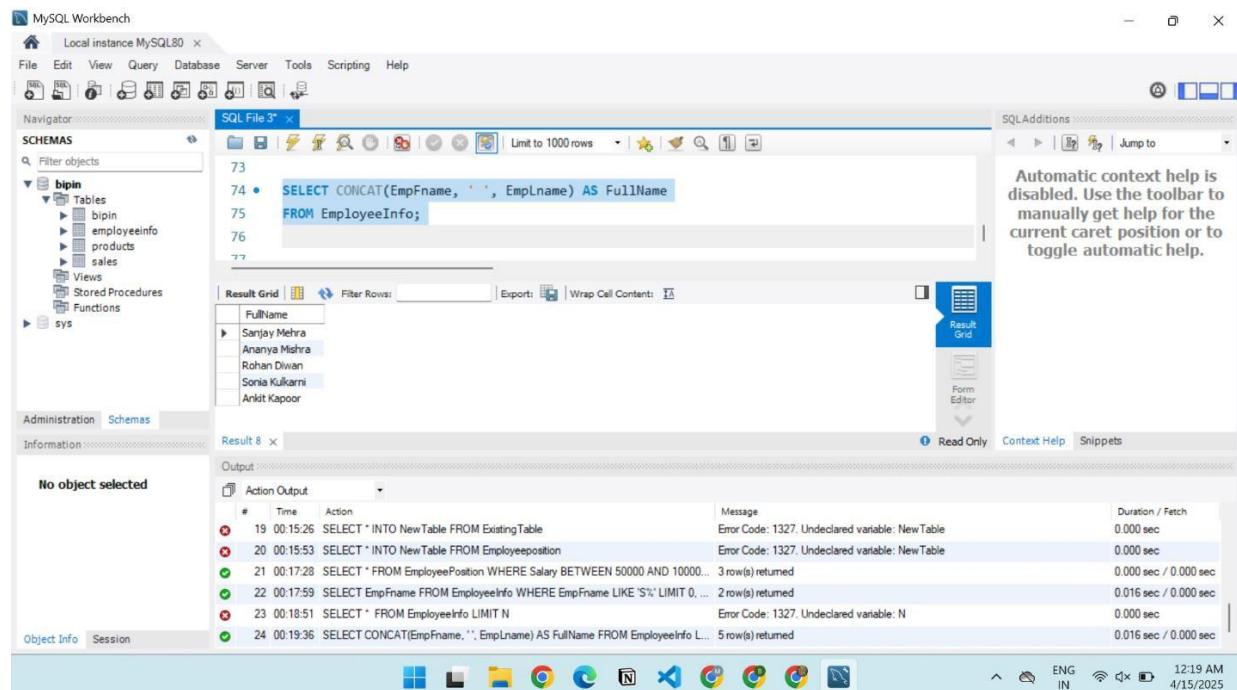
The Result Grid shows the following data:

EmpID	EmpName	Employee	Department	Project	Address	DOB	Gender
1	Sanjay	Mehra	HR	Hyderabad(HD)	1978-12-01	M	
2	Ananya	Mishra	Admin	P2	1988-05-02	F	
3	Rohan	Diwan	Account	P3	Mumbai(BOM)	1990-01-01	M

The Output pane shows the execution history:

- 20 09:03:52 SELECT COUNT(\*) AS HR\_Employee\_Count FROM EmployeeInfo WHERE Department = 'HR' LIMIT 0,1000
- 21 09:04:25 SELECT CURRENT\_DATE LIMIT 0,1000
- 22 09:04:57 SELECT SUBSTRING(EmpName, 1, 4)AS ShortName FROM EmployeeInfo LIMIT 0,1000
- 23 09:05:32 CREATE TABLE NewEmployeeInfo AS SELECT \* FROM EmployeeInfo
- 24 09:05:39 select \* from NewEmployeeInfo LIMIT 0,1000
- 25 09:06:20 SELECT SUBSTRING\_INDEX(Address, ',', 1)AS Place FROM EmployeeInfo LIMIT 0,1000
- 26 09:07:14 SELECT \* FROM EmployeePosition WHERE Salary BETWEEN 50000 AND 100000 LIMIT 0,1000
- 27 09:07:40 SELECT EmpName FROM EmployeeInfo WHERE EmpName LIKE '%N%' LIMIT 0,1000
- 28 09:08:07 SELECT \* FROM EmployeeInfo LIMIT 3

## 10. Write a query to retrieve the EmpFname and EmpLname in a single column as “FullName”. The first name and the last name must be separated with space.



The screenshot shows the MySQL Workbench interface with a SQL editor containing the following query:

```

73
74 • SELECT CONCAT(EmpFname, ' ', EmpLname) AS FullName
75 FROM EmployeeInfo;
76
77
```

The Result Grid shows the following data:

FullName
Sanjay Mehra
Ananya Mishra
Rohan Diwan
Sonia Kukarni
Ankit Kapoor

The Output pane shows the execution history:

- 19 00:15:26 SELECT \* INTO NewTable FROM ExistingTable
- 20 00:15:53 SELECT \* INTO NewTable FROM EmployeePosition
- 21 00:17:28 SELECT \* FROM EmployeePosition WHERE Salary BETWEEN 50000 AND 10000...
- 22 00:17:59 SELECT EmpName FROM EmployeeInfo WHERE EmpName LIKE '%N%' LIMIT 0, ...
- 23 00:18:51 SELECT \* FROM EmployeeInfo LIMIT N
- 24 00:19:36 SELECT CONCAT(EmpFname, ' ', EmpLname) AS FullName FROM EmployeeInfo L...

A tooltip in the right panel states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

## 8.To implement Joins (inner, outer, natural, equi join) and related example.

Joins in SQL are used to combine rows from two or more tables based on a related column between them. Here are the common types of joins:

```
mysql> select * from a1;
+----+-----+
| id | name |
+----+-----+
| 1 | akash_rattan |
| 2 | a |
| 3 | b |
| 4 | c |
+----+-----+
4 rows in set (0.00 sec)

mysql> select * from a2;
+----+-----+
| id | city |
+----+-----+
| 1 | noida |
| 2 | delhi |
| 3 | lucknow |
+----+-----+
```

### Inner Join

An inner join returns rows when there is at least one match in both tables.

```
mysql> select a1.name,a2.city from a1 inner join a2 on a1.id=a2.id;
+-----+-----+
| name | city |
+-----+-----+
| akash_rattan | noida |
| a | delhi |
| b | lucknow |
+-----+-----+
```

### Left Outer Join

A left outer join returns all rows from the left table (table1), and the matched rows from the right table (table2). The result is NULL from the right side, if there is no match.

```
mysql> select a1.name,a2.city from a1 left join a2 on a1.id=a2.id;
+-----+-----+
| name | city |
+-----+-----+
| akash_rattan | noida |
| a | delhi |
| b | lucknow |
| c | NULL |
+-----+-----+
```

### Right Outer Join

A right outer join returns all rows from the right table (table2), and the matched rows from the left table (table1). The result is NULL from the left side, when there is no match.

```
mysql> select a1.name,a2.city from a1 right join a2 on a1.id=a2.id;
+-----+-----+
| name | city |
+-----+-----+
| akash_rattan | noida |
| a | delhi |
| b | lucknow |
+-----+-----+
```

## Full Outer Join

A full outer join returns all records when there is a match in either left (table1) or right (table2) table records.

```
mysql> select a1.name,a2.city from a1 left join a2 on a1.id=a2.id
 -> union
 -> select a1.name,a2.city from a1 right join a2 on a1.id=a2.id;
+-----+-----+
| name | city |
+-----+-----+
| akash | noida |
| a | delhi |
| b | lucknow |
| c | NULL |
+-----+-----+
```

## Natural Join

A natural join is based on all columns in the two tables that have the same name and selects rows with equal values in those columns.

```
mysql> select * from a1 natural join a2;
+-----+-----+-----+
| id | name | city |
+-----+-----+-----+
| 1 | akash_rattan | noida |
| 2 | a | delhi |
| 3 | b | lucknow |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

## Equi Join

An equi join is a specific type of join that combines tables based on matching values in specified columns.

```
mysql> select a1.name,a2.city from a1 ,a2 where a1.id=a2.id;
+-----+-----+
| name | city |
+-----+-----+
| akash_rattan | noida |
| a | delhi |
| b | lucknow |
+-----+-----+
```

## 9. Introduction of VIEW. How to create view. Explain with example.

In the context of databases, a view is a virtual table that consists of a subset of data from one or more tables. Views are useful for various purposes, such as simplifying complex queries, providing a layer of security by restricting access to certain columns or rows, and presenting data in a particular format for reporting or analysis.

### Creating a view in a typical MySQL database -

```
```CREATE VIEW view_name AS
```

```
SELECT column1, column2,
```

```
FROM table_name
```

```
WHERE condition;```
```

```
mysql> select * from new;
+---+-----+---+
| id | name      | age |
+---+-----+---+
| 1  | akash_rattan | 56 |
| 2  | oggy       | 30 |
| 3  | jack        | 10 |
| 4  | panther    | 5  |
+---+-----+---+
4 rows in set (0.00 sec)

mysql> create view second as select name,age from new where age>30;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from second;
+-----+---+
| name      | age |
+-----+---+
| akash_rattan | 56 |
+-----+---+
1 row in set (0.00 sec)
```