Experiment No.6
Implement various join operations
Date of Performance:
Date of Submission:

Aim :- Write simple query to implement join operations(equi join, natural join, inner join, outer joins).

Objective :- To apply different types of join to retrieve queries from the database management system.

Theory: SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

A. INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching_column = table2.matching_column;
```

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table

matching_column: Column common to both the tables.

C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

FULL JOIN table2

ON table1.matching_column = table2.matching_column;

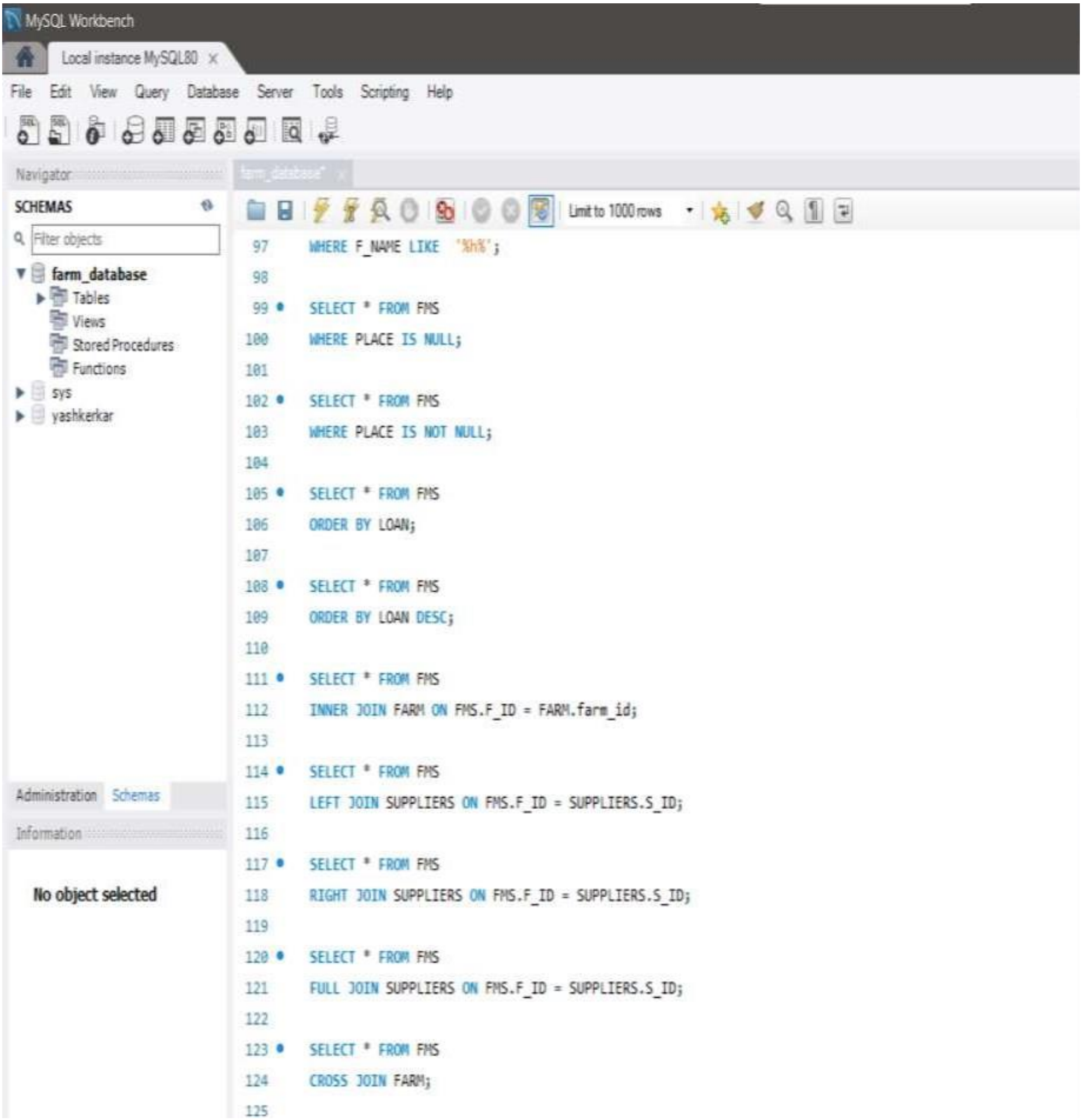
table1: First table.

table2: Second table

matching_column: Column common to both the tables.

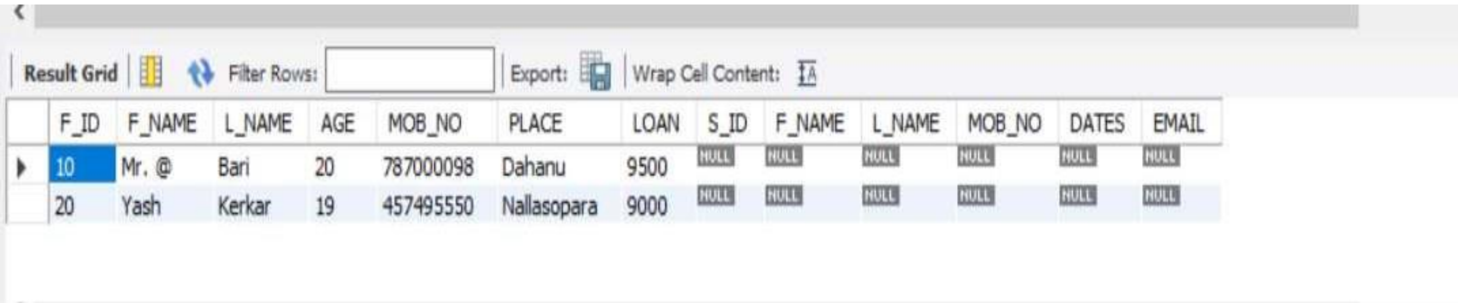
Implementation:

Code:

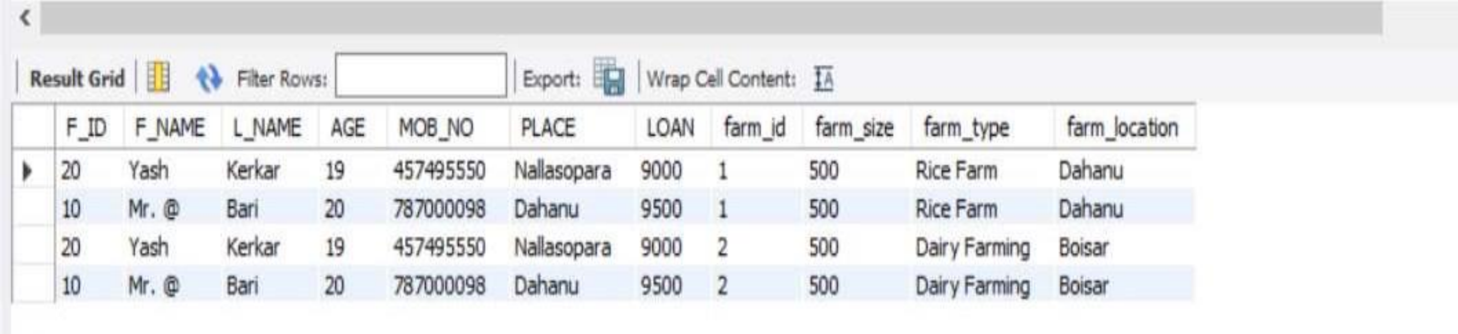


```
WHERE F_NAME LIKE '%h%';

SELECT * FROM FMS
WHERE PLACE IS NULL;

SELECT * FROM FMS
WHERE PLACE IS NOT NULL;

SELECT * FROM FMS
ORDER BY LOAN;

SELECT * FROM FMS
ORDER BY LOAN DESC;

SELECT * FROM FMS
INNER JOIN FARM ON FMS.F_ID = FARM.farm_id;

SELECT * FROM FMS
LEFT JOIN SUPPLIERS ON FMS.F_ID = SUPPLIERS.S_ID;

SELECT * FROM FMS
RIGHT JOIN SUPPLIERS ON FMS.F_ID = SUPPLIERS.S_ID;

SELECT * FROM FMS
FULL JOIN SUPPLIERS ON FMS.F_ID = SUPPLIERS.S_ID;

SELECT * FROM FMS
CROSS JOIN FARM;
```

Output:



	F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN	S_ID	F_NAME	L_NAME	MOB_NO	DATES	EMAIL
▸	10	Mr. @	Bari	20	787000098	Dahanu	9500	NULL	NULL	NULL	NULL	NULL	NULL
	20	Yash	Kerkar	19	457495550	Nallasopara	9000	NULL	NULL	NULL	NULL	NULL	NULL



	F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN	farm_id	farm_size	farm_type	farm_location
▸	20	Yash	Kerkar	19	457495550	Nallasopara	9000	1	500	Rice Farm	Dahanu
	10	Mr. @	Bari	20	787000098	Dahanu	9500	1	500	Rice Farm	Dahanu
	20	Yash	Kerkar	19	457495550	Nallasopara	9000	2	500	Dairy Farming	Boisar
	10	Mr. @	Bari	20	787000098	Dahanu	9500	2	500	Dairy Farming	Boisar

Conclusion:

In this practical exercise, we successfully implemented various join operations to combine data from different tables within the FARM_DATABASE. By utilizing different types of joins, including INNER JOIN, RIGHT JOIN, LEFT JOIN, and CROSS JOIN, we gained insights into how data from multiple tables can be merged based on common attributes.

1. Illustrate how to perform natural join for the joining attributes with different names with a suitable example.

Ans. CREATE TABLE FMS (

```
    F_ID INT PRIMARY KEY,

    F_NAME VARCHAR(25),

    L_NAME VARCHAR(25),

    AGE INT,

    PLACE VARCHAR(50)

);


CREATE TABLE FARM (

    farm_id INT PRIMARY KEY,
    farm_size INT,
    farm_type VARCHAR(50),
    farm_location VARCHAR(50)
);
```

Now, let's say we want to perform a natural join between these tables to find farmers along with their respective farms. The columns representing the location in both tables have different names (**PLACE** in **FMS** and **farm_location** in **FARM**). We'll use aliases to match these columns:

```
SELECT *
FROM FMS
NATURAL JOIN FARM;
```

2. Illustrate significant differences between natural join equi join and inner join

Ans. Natural Join:

- A natural join is a type of join that automatically matches columns with the same name in the two tables being joined.
- It does not require specifying the join condition explicitly.
- The result set includes only those rows where the values in the matching columns are equal.
- If the columns have different names but represent the same data, you can use aliases to match them.
- Natural joins can result in unintended matches if there are columns with the same name but different meanings.

2. **Equi Join**:

- An equi join is a type of join that explicitly specifies the equality condition between columns from two tables.
- It uses the = operator to match values in the specified columns.
- Equi joins can involve columns with different names or columns with the same name but different meanings.
- You must explicitly specify the join condition using the **ON** keyword.
- The result set includes only those rows where the values in the specified columns are equal.

3. **Inner Join**:

- An inner join is a type of join that returns only the rows from both tables that satisfy the join condition.
- It can be either a natural join or an equi join, depending on how the join condition is specified.
- If no join condition is specified explicitly, an inner join acts as a natural join, matching columns with the same name.
- Inner joins are versatile and widely used for combining data from multiple tables based on specified conditions.
- You must explicitly specify the join condition using the **ON** keyword in the case of an equi join.