Experiment No.5
Perform simple queries, string manipulation operations and aggregate functions.
Date of Performance:
Date of Submission:

Aim :- Write simple query to manipulate string operations and perform aggregate functions like (MIN, MAX, SUM, AVERAGE, COUNT).

Objective :- To apply aggregate functions and string manipulation functions to perform simple queries in the database system

Theory:

Simple Queries in SQL:

In SQL, a simple query is a request for data from a database table or tables. It allows users to retrieve specific information by specifying the columns they want to retrieve and any conditions for filtering rows based on certain criteria. Simple queries are the backbone of interacting with databases, enabling users to extract the data they need for analysis, reporting, or further processing.

String Manipulation Operations:

String manipulation operations in SQL involve modifying or transforming string values stored in database columns. These operations are crucial for tasks such as formatting data, combining strings, converting case, or extracting substrings. By using string functions and operators, users can manipulate text data to suit their requirements, whether it's for display purposes or for further analysis.

Aggregate Functions:

Aggregate functions in SQL are used to perform calculations on sets of values and return a single result. These functions allow users to summarize data across multiple rows, providing insights into the overall characteristics of the dataset. Common aggregate functions include calculating counts, sums, averages, minimums, and maximums of numerical values. They are essential tools for data analysis, enabling users to derive meaningful insights from large datasets.

Benefits of Understanding These Concepts:

● Data Retrieval: Simple queries allow users to fetch specific data from databases,

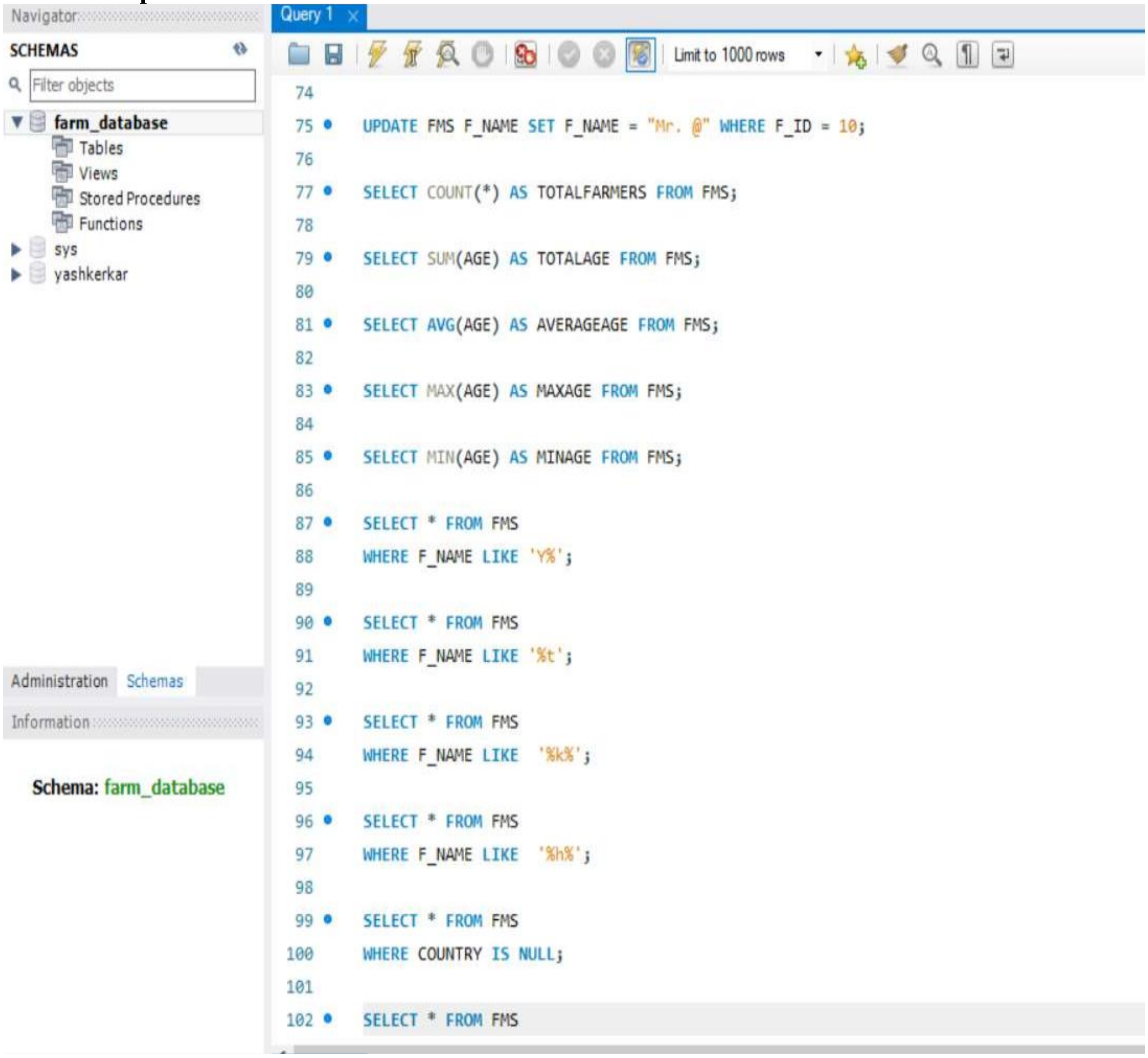
facilitating data retrieval for various purposes.

● Data Transformation: String manipulation operations enable users to format and transform text data according to their needs, improving data consistency and readability.

● Data Analysis: Aggregate functions help users summarize and analyze large datasets, providing valuable insights into trends, patterns, and statistical measures.

● Data Reporting: By combining simple queries, string manipulation operations, and aggregate functions, users can generate reports and visualizations that communicate key findings effectively.

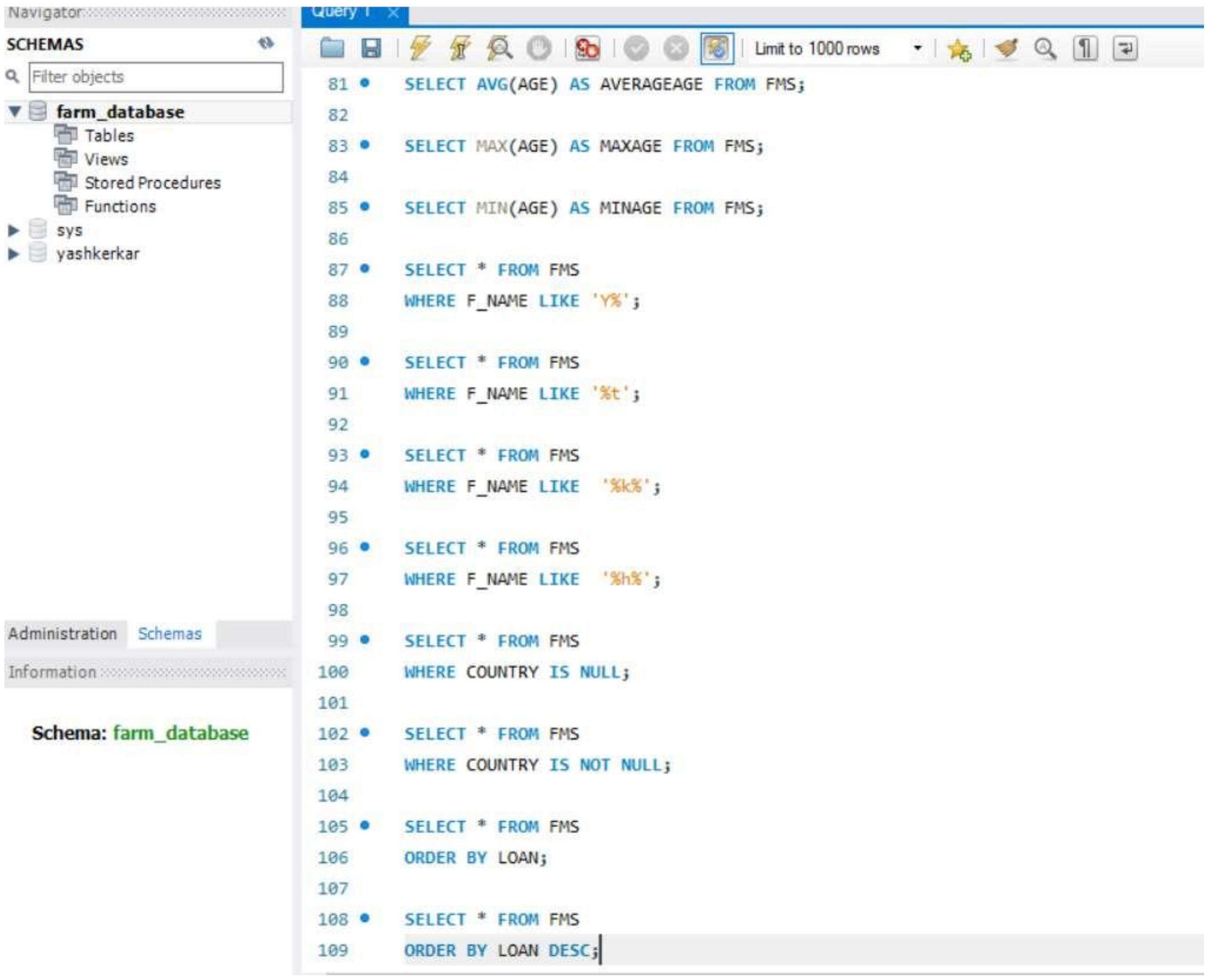
Implementation:



```sql
UPDATE FMS F_NAME SET F_NAME = "Mr. @" WHERE F_ID = 10;

SELECT COUNT(*) AS TOTALFARMERS FROM FMS;

SELECT SUM(AGE) AS TOTALAGE FROM FMS;

SELECT AVG(AGE) AS AVERAGEAGE FROM FMS;

SELECT MAX(AGE) AS MAXAGE FROM FMS;

SELECT MIN(AGE) AS MINAGE FROM FMS;

SELECT * FROM FMS
WHERE F_NAME LIKE 'Y%';

SELECT * FROM FMS
WHERE F_NAME LIKE '%t';

SELECT * FROM FMS
WHERE F_NAME LIKE  '%k%';

SELECT * FROM FMS
WHERE F_NAME LIKE  '%h%';

SELECT * FROM FMS
WHERE COUNTRY IS NULL;

SELECT * FROM FMS
```

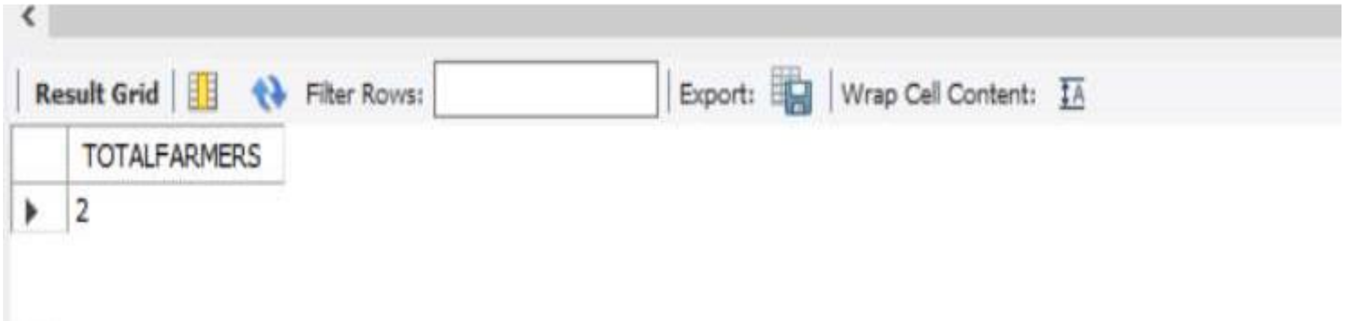
Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

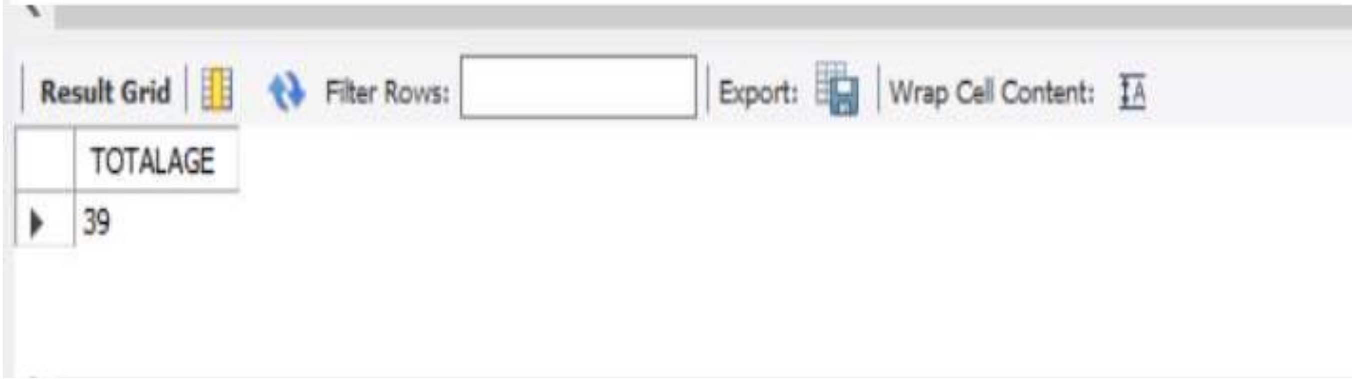


```
SELECT AVG(AGE) AS AVERAGEAGE FROM FMS;

SELECT MAX(AGE) AS MAXAGE FROM FMS;

SELECT MIN(AGE) AS MINAGE FROM FMS;

SELECT * FROM FMS
WHERE F_NAME LIKE 'Y%';

SELECT * FROM FMS
WHERE F_NAME LIKE '%t';

SELECT * FROM FMS
WHERE F_NAME LIKE  '%k%';

SELECT * FROM FMS
WHERE F_NAME LIKE  '%h%';

SELECT * FROM FMS
WHERE COUNTRY IS NULL;

SELECT * FROM FMS
WHERE COUNTRY IS NOT NULL;

SELECT * FROM FMS
ORDER BY LOAN;

SELECT * FROM FMS
ORDER BY LOAN DESC;
```

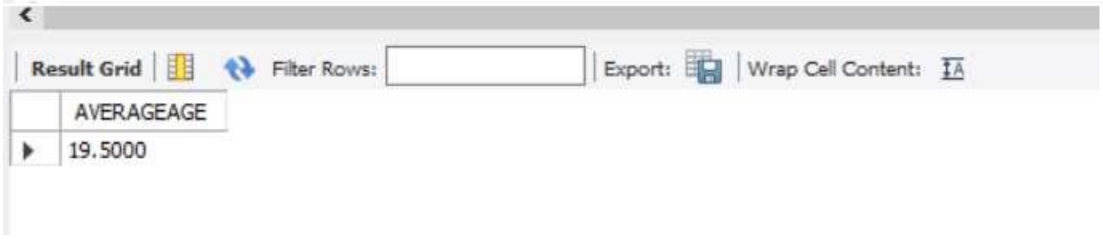
Output:



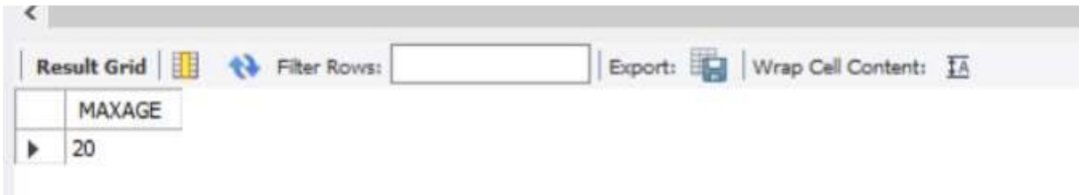
	TOTALFARMERS
▶	2



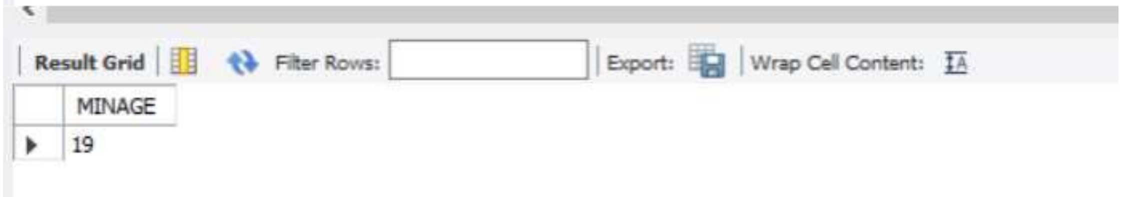
	TOTALAGE
▶	39



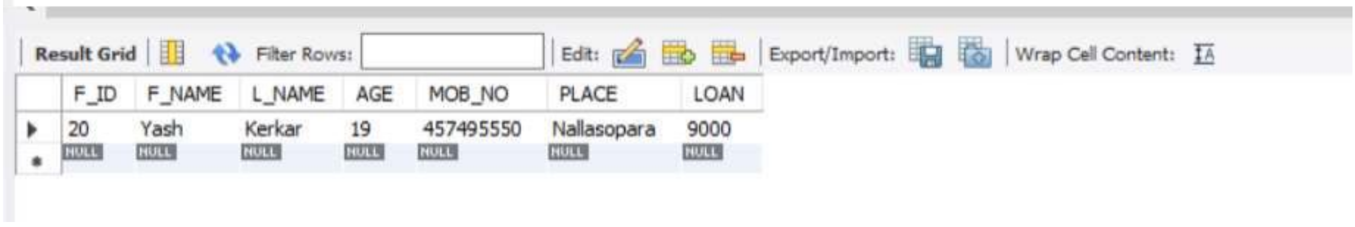
	AVERAGEAGE
▶	19.5000



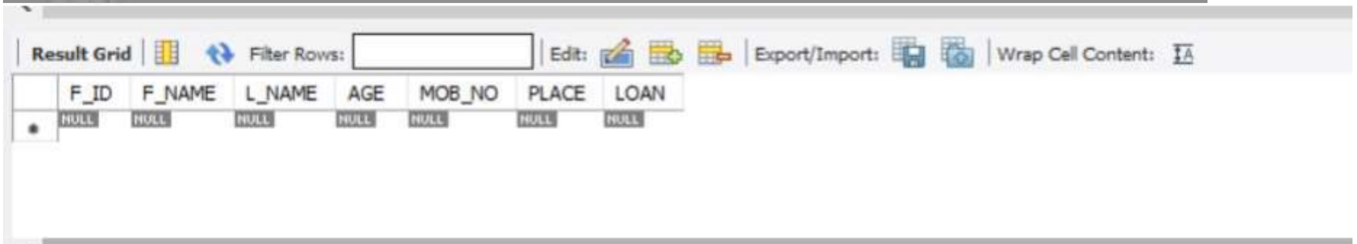
	MAXAGE
▶	20



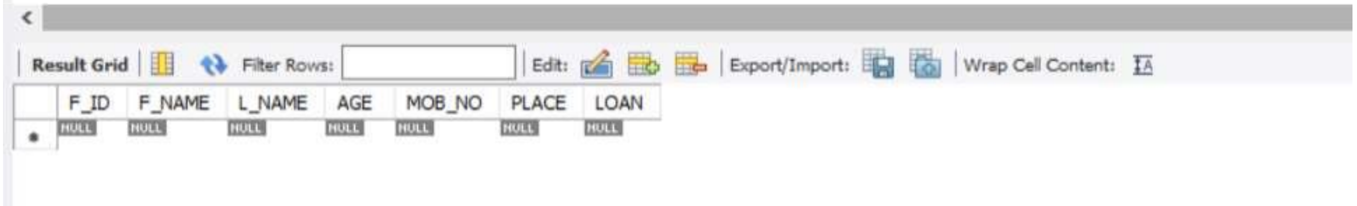
	MINAGE
▶	19



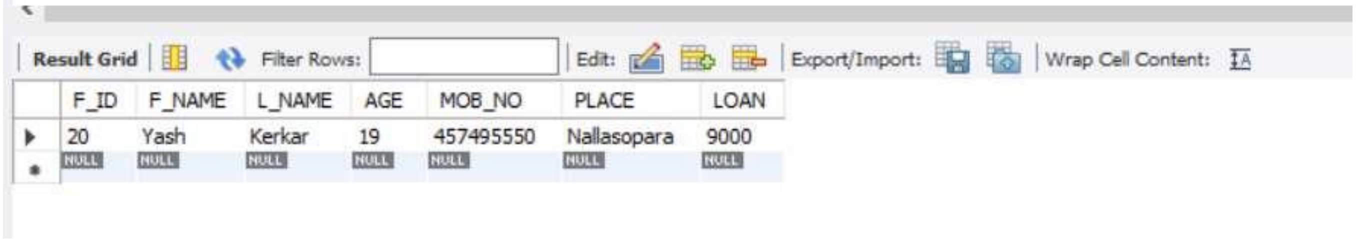
	F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN
▶	20	Yash	Kerkar	19	457495550	Nallasopara	9000
●	NULL	NULL	NULL	NULL	NULL	NULL	NULL



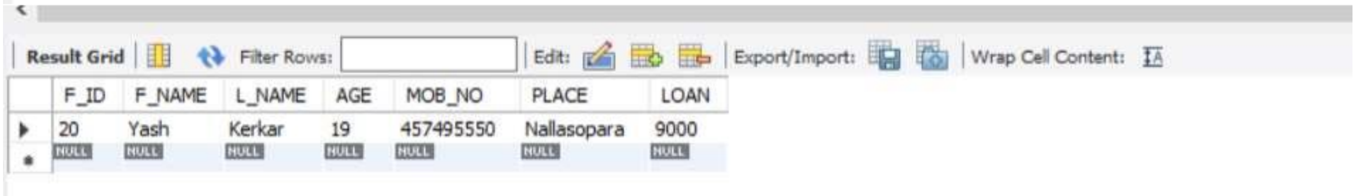
F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN
NULL	NULL	NULL	NULL	NULL	NULL	NULL



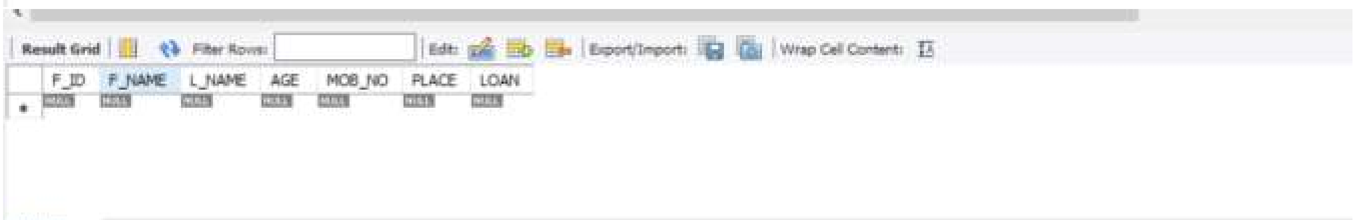
F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN
NULL	NULL	NULL	NULL	NULL	NULL	NULL



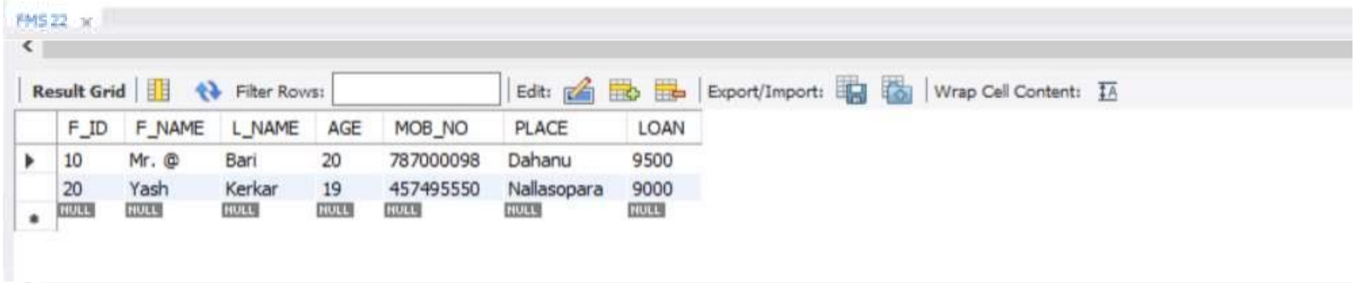
F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN
20	Yash	Kerkar	19	457495550	Nallasopara	9000
NULL	NULL	NULL	NULL	NULL	NULL	NULL



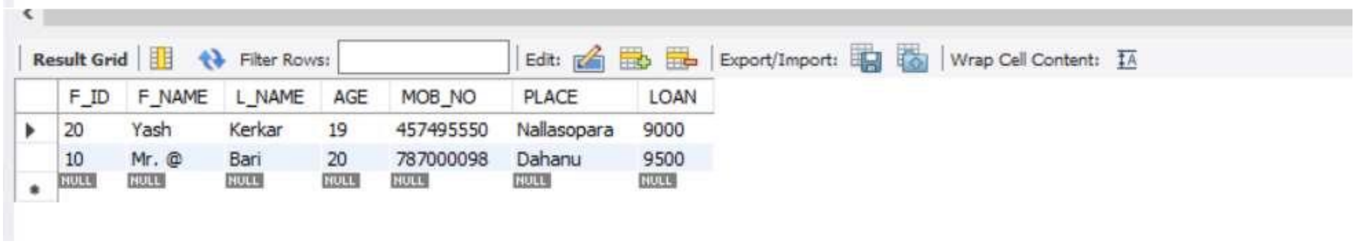
F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN
20	Yash	Kerkar	19	457495550	Nallasopara	9000
NULL	NULL	NULL	NULL	NULL	NULL	NULL



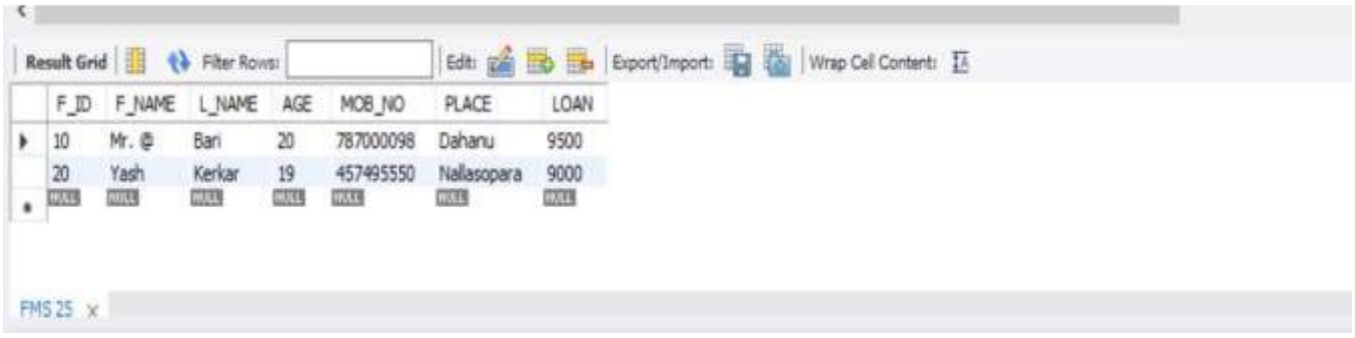
F_ID	F_NAME	L_NAME	AGE	MOB_NO	PLACE	LOAN
NULL	NULL	NULL	NULL	NULL	NULL	NULL



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



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



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

FMS 25 ×

Conclusion:

Through this practical exercise, we have successfully demonstrated the proficiency in performing various queries, string manipulation operations, and aggregate functions in a database management system. These operations are fundamental in managing and extracting meaningful insights from data stored within databases.

1. Write syntax and explanation for each of the five aggregate functions

1. Ans. COUNT():
   - Syntax: **COUNT(expression)**
   - Explanation: This function counts the number of rows in a specified column or all rows in a table if no column is specified. It ignores NULL values unless the expression is **COUNT(*)**, which counts all rows regardless of NULL values.
2. **SUM()**:
   - Syntax: **SUM(expression)**
   - Explanation: This function calculates the sum of all non-NULL values in a specified column. It is commonly used with numeric data types such as integers, decimals, or floating-point numbers.
3. **AVG()**:
   - Syntax: **AVG(expression)**
   - Explanation: This function calculates the average (mean) of all non-NULL values in a specified column. It is useful for finding the typical value within a dataset and is applicable to numeric data types.
4. **MIN()**:
   - Syntax: **MIN(expression)**
   - Explanation: This function returns the smallest (minimum) value in a specified column. It is typically used with numeric, string, or date/time data types to find the lowest value within a dataset.

5. **MAX()**:
    - Syntax: **MAX(expression)**
    - Explanation: This function returns the largest (maximum) value in a specified column. Similar to MIN(), it is applicable to numeric, string, or date/time data types and is used to find the highest value within a dataset.

2. Show results of operations performed.

Ans. **COUNT()**:
```
SELECT COUNT(*) AS total_farmers FROM
fms; total_farmers
------------
 --50
```

SUM():
```
SELECT SUM(loan) AS total_loans FROM
 FMS; total_loans
 ---------
 --
 12500
```

AVG():
```
SELECT AVG(loan) AS average_loan FROM
FMS; average_salary
-------------
--55000
```

MIN():
```
SELECT MIN(age) AS lowest_age FROM
 FMS; lowest_age
 ----------
 --19
```

MAX():
```
        SELECT MAX(age) AS highest_age_date FROM
        FMS; lowest_age
------------19
```