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:

CONCAT:-

```
Limit to 1000 rows
 9  INSERT INTO users (user_id, username, email, password)
10  VALUES ('107', 'Aaru', 'aradhya@gmail.com', 'wewy2vduqw');
11  INSERT INTO users (user_id, username, email, password)
12  VALUES ('108', 'Ansh', 'ansha@gmail.com', 'wewqwy2vduqw');
13  INSERT INTO users (user_id, username, email, password)
14  VALUES ('109', 'Luv', 'luv@gmail.com', 'wadewy2vduqwwr4c');
15  INSERT INTO users (user_id, username, email, password)
16  VALUES ('110', 'Kush', 'kush@gmail.com', 'wedsdwy2vduqw');
17  INSERT INTO users (user_id, username, email, password)
18  VALUES ('111', 'Ram', 'ram@gmail.com', 'wwy2vduk;qw');
19  INSERT INTO users (user_id, username, email, password)
20  VALUES ('112', 'Sans', 'sanskruti@gmail.com', 'wewy2vggduqw');
21
22  USE social_media_db;
23  SELECT UPPER(username) FROM users WHERE user_id = '112';
24
25  SELECT CONCAT ('Luv','Kush') from users;
```

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

CONCAT ('Luv','Kush')
LuvKush
LuvKush
LuvKush
LuvKush
LuvKush
LuvKush

LENGTH:-

```
Limit to 1000 rows
 8  );
 9  INSERT INTO users (user_id, username, email, password)
10  VALUES ('107', 'Aaru', 'aradhya@gmail.com', 'wewy2vduqw');
11  INSERT INTO users (user_id, username, email, password)
12  VALUES ('108', 'Ansh', 'ansha@gmail.com', 'wewqwy2vduqw');
13  INSERT INTO users (user_id, username, email, password)
14  VALUES ('109', 'Luv', 'luv@gmail.com', 'wadewy2vduqwwr4c');
15  INSERT INTO users (user_id, username, email, password)
16  VALUES ('110', 'Kush', 'kush@gmail.com', 'wedsdwy2vduqw');
17  INSERT INTO users (user_id, username, email, password)
18  VALUES ('111', 'Ram', 'ram@gmail.com', 'wwy2vduk;qw');
19  INSERT INTO users (user_id, username, email, password)
20  VALUES ('112', 'Sans', 'sanskruti@gmail.com', 'wewy2vggduqw');
21
22  select username,length(username) from users;
23
24
```

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

username	length(username)
Aaru	4
Ekansh	6
Kush	4
Luv	3
Parnu	5
Sans	4

LOWER:-

```
14      VALUES ('109', 'Luv', 'luv@gmail.com', 'wadewy2vduqwwr4c');
15 •    INSERT INTO users (user_id, username, email, password)
16      VALUES ('110', 'Kush', 'kush@gmail.com', 'wedsdwy2vduqw');
17 •    INSERT INTO users (user_id, username, email, password)
18      VALUES ('111', 'Ram', 'ram@gmail.com', 'wwy2vduk;qw');
19 •    INSERT INTO users (user_id, username, email, password)
20      VALUES ('112', 'Sans', 'sanskruti@gmail.com', 'wewy2vggduqw');
21
22 •    USE social_media_db;
23 •    SELECT LOWER(username) FROM users WHERE user_id = '112';
24
25
26
27
```

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

LOWER(username)
sans

REPLACE :-

```
1
2 •    USE social_media_db;
3 •    SELECT UPPER(username) FROM users WHERE user_id = '112';
4
5 •    SELECT REPLACE (username,'Parnu','Konisha') As username
6      FROM users;
```

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

username
Aaru
Ekansh
Kush
Luv
Konisha
Sans

SUBSTRING :-

```
1
2 •  USE social_media_db;
3 •  SELECT UPPER(username) FROM users WHERE user_id = '112';
4
5 •  SELECT SUBSTRING(user_id, 2, 3) AS username
6    FROM users;
```

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

username
07
08
10
09
13
12

UPPER:-

```
11 •  INSERT INTO users (user_id, username, email, password)
12    VALUES ('108', 'Ansh', 'ansha@gmail.com', 'wewqwy2vduqw');
13 •  INSERT INTO users (user_id, username, email, password)
14    VALUES ('109', 'Luv', 'luv@gmail.com', 'wadewy2vduqwwr4c');
15 •  INSERT INTO users (user_id, username, email, password)
16    VALUES ('110', 'Kush', 'kush@gmail.com', 'wedsdwy2vduqw');
17 •  INSERT INTO users (user_id, username, email, password)
18    VALUES ('111', 'Ram', 'ram@gmail.com', 'wwy2vduk;qw');
19 •  INSERT INTO users (user_id, username, email, password)
20    VALUES ('112', 'Sans', 'sanskruti@gmail.com', 'wewy2vggduqw');
21
22 •  USE social_media_db;
23 •  SELECT UPPER(username) FROM users WHERE user_id = '112';
24
25
26
27
```

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

UPPER(username)
SANS

Conclusion:

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

a) COUNT():

Syntax: COUNT(expression)

Explanation: Counts the number of rows in a table or the number of non-null values in a specific column. The expression can be either an asterisk * to count all rows, or a specific column name to count non-null values in that column.

b) SUM():

Syntax: SUM(expression)

Explanation: Calculates the sum of all values in a numeric column. The expression must reference a numeric column. NULL values are ignored in the calculation.

c) AVG():

Syntax: AVG(expression)

Explanation: Calculates the average (mean) value of all values in a numeric column. The expression must reference a numeric column. NULL values are ignored in the calculation.

d) MIN():

Syntax: MIN(expression)

Explanation: Returns the minimum value in a column. The expression must reference a column containing values that can be compared. NULL values are ignored in the comparison.

e) MAX():

Syntax: MAX(expression)

Explanation: Returns the maximum value in a column. The expression must reference a column containing values that can be compared. NULL values are ignored in the comparison.

2. Show results of operations performed.

COUNT():
SELECT COUNT(*) FROM Sales;
Result: 5 (Counts all rows in the table)

SUM():
SELECT SUM(Quantity) FROM Sales;
Result: 55 (Sum of all non-null values in the Quantity column)

AVG():
SELECT AVG(Price) FROM Sales;
Result: 6 (Average of all non-null values in the Price column)

MIN():
SELECT MIN(Price) FROM Sales;
Result: 4 (Minimum value in the Price column)

MAX():
SELECT MAX(Quantity) FROM Sales;
Result: 20 (Maximum value in the Quantity column)