**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**DATA BASE MANAGEMENT SYSTEM**

**PRACTICAL FILE**

**Submitted to :**

**Dr . Prerna Sharma**

**Submitted By:**

Name: Ridham

Sec:4C-9

Roll no :75114802719

**DATA BASE MANAGEMENT SYSTEM**

**EXPERIMENT :5**

**Aim: Write the queries for implementing the following functions:MAX(),MIN(), AVG(), and logical pattern matching operations.**

**Tools Used:** Maria Db

**Theory and Prodecure:**

In this experiment we will be working on different operations to be applied on a table . These operations include pattern matching operations like ‘LIKE’, arithmetic operations like +,-,/,\*,% and logical operations like AND, BETWEEN ,IN ,OR,NOT EXISTS etc.

**SQL Like Operator:**

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. There are two wildcards used in conjunction with the LIKE operator:

% 🡪 The percentage sign represents zero,one or multiple characters.

\_ 🡪 The underscore represents a single character.

**Syntax:**

SELECT column1,column2,…

FROM table\_name

Where column LIKE pattern;

|  |  |
| --- | --- |
| LIKE Operator | DESCRIPTION |
| WHERE CustomerName LIKE ‘a%’ | Finds any value that starts with “a” |
| WHERE CustomerName LIKE ‘%a’ | Finds any value that ends with “a” |
| WHERE CustomerName LIKE ‘%or%’ | Finds any value that has “or” in any position |
| WHERE CustomerName LIKE ‘\_r%’ | Finds any value that has “r” in the second position. |
| WHERE CustomerName LIKE ‘a\_%\_%’ | Finds any value that starts with “a” and are at least 3 characters in length. |
| WHERE ContactName LIKE ‘a%o’ | Finds any value that starts with “a” and ends with “o”. |

**SQL Arithmetic Operators:**

|  |  |
| --- | --- |
| **OPERATOR** | **DESCRIPTION** |
| **+** | Add |
| **-** | Subtract |
| **\*** | Multiply |
| / | Divide |
| % | Modulo |

**SQL Logical Operators:**

|  |  |
| --- | --- |
| **OPERATOR** | **DESCRIPTION** |
| ALL | True if all of the subquery values meet the condition |
| AND | True if all the conditions separated by AND is true. |
| ANY | True if any of the subquery meets the condition. |
| BETWEEN | True if operand is within the range of comparisons. |
| EXISTS | True if the subquery returns one or more records. |
| IN | True if the operand is equal to one of a list of expressions. |
| NOT | Displays a record if the condition(s) is NOT TRUE |
| OR | True if any of the conditions separated by OR is true. |

**SOME SQL OPERATORS AND THEIR SYNATAX**

1. **ALL Operator:** the all operator returns true if all of the subquery values meet the condition.

SYNTAX:

Select column\_name

From tablename

WHERE column\_name operator ALL

(SELECT column\_name From table\_name WHERE condition);

1. **ANY Operator:** the any operator returns true if any of the subqueries values meet the condition

SYNTAX:

Select columnname

From tablename

WHERE column\_name operator ANY

(SELECT column\_name From table\_name WHERE condition);

1. **THE SQL AND ,OR and NOT operator:**

**AND**: The AND operator display the record if all the conditions separated by AND are TRUE

SYNTAX: Select column1,column2,..

From tablename

WHERE condition1 AND condition2 AND condition3……;

**OR :** The OR operator displays a record if any of the conditions separated by OR are TRUE.

SYNTAX: Select column1,column2,..

From tablename

WHERE condition1 OR condition2 ……;

**NOT:** The NOT operator displays a record if the condition is NOT TRUE.

SYNTAX: Select column1,column2,..

From tablename

WHERE NOT condition;

1. **BETWEEN Operator**: It selects values within a given range. the values can be numbers, text, or dates

**SYNTAX:** SELECT columnname

FROM tablename

WHERE columnname BETWEEN value 1 and value2.

1. **EXISTS Operator:** The EXISTS is used to test for the existence of any record in a subquery. It returns TRUE if the subquery returns one or more records.

**SYNTAX:** SELECT columnname

FROM tablename WHERE EXISTS

(SELECT column\_name from table\_name WHERE condition);

1. **IN Operator:** this allows you to specify multiple values in a WHERE clause. It is a shorthand for multiple OR conditions.

**SYNTAX:** SELECT columnname

FROM tablename

WHERE columnname IN (value1,value 2….);

1. **MIN() Operator:** This function returns the smallest value of the selected column

**SYNTAX:** SELECT MIN columnname

FROM tablename

WHERE condition

1. **MAX() Operator:** This function return store largest value all the selector column

**SYNTAX:** SELECT MAX columnname

FROM tablename

WHERE condition

1. **COUNT() Operator:** This function return store number of rows that matches a specified criteria

**SYNTAX:** SELECT COUNT (columnname )

FROM tablename

WHERE condition

1. **AVG() Operator:** This function returns the average value of numeric column

**SYNTAX:** SELECT AVG (columnname )

FROM tablename

WHERE condition

1. **SUM() Operator:** This function returns the total sum of numatic column

**SYNTAX:** SELECT SUM (columnname )

FROM tablename

WHERE condition

**QUERIES :**

1. **List the names of all clients having ‘a’ as the 2nd letter in their names.**

**Input Query:**

**SELECT** **NAME**

**FROM** client\_master

**WHERE** **NAME** **LIKE** '\_a%';

**Input:**

**A picture containing graphical user interface

Description automatically generated**

**Output:**

Application

Description automatically generated with low confidence

1. **List the client who stay in the city whose first letter is ‘M’.**

**Input Query:**

**SELECT** \***FROM** client\_master

**WHERE** **NAME** **LIKE** 'M%';

**Input:**

**Background pattern

Description automatically generated with low confidence**

**Output:**

Graphical user interface, application

Description automatically generated3

1. **List all clients who stay in ‘Bangalore’ or ‘Mangalore’**

**Input Query:**

**SELECT** \***FROM** client\_master

**WHERE** city ='Bangalore' **OR** city ='Mangalore';

**Input:**

**A picture containing text

Description automatically generated**

**Output:**

Graphical user interface, application

Description automatically generated

1. **List all the clients whose bal\_due equals 10000.**

**Input Query:**

**SELECT** \* **FROM** client\_master

**WHERE** baldue >=10000;

**Input:**

**Background pattern

Description automatically generated with low confidence**

**Output:**

Graphical user interface, application

Description automatically generated

1. **List all the information sales\_order for orders places in the month of JUNE.**

**Input Query:**

**SELECT** \* **FROM** sales\_order

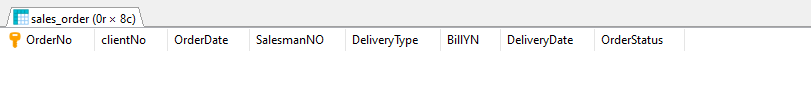
**WHERE** orderdate **BETWEEN** 01-06-2020 **AND** 30-06-2025;

**Input:**

**A picture containing text

Description automatically generated**

**Output:**



1. **List the order information for the client number lies between ‘C00001’and ‘C00002’**

**Input Query:**

**SELECT** \* **FROM** client\_master

**WHERE** client\_no='C00001' **OR** client\_no='C00002';

**Input:**

**Text

Description automatically generated**

**Output:**

Graphical user interface, table

Description automatically generated with medium confidence

1. **List the products whose selling price is greater than 500 and less than or equal to 750.**

**Input Query:**

**SELECT** description

**FROM** product\_master1

**WHERE** sellprice **BETWEEN** 500 **AND** 750;

**Input:**

**Logo

Description automatically generated**

**Output:**

Graphical user interface, text, application

Description automatically generated

1. **List products whose SP is more than 500. Calculate the new SP as original SP\*0.15.Rename the new column in the output as new\_price.**

**Input Query:**

**SELECT** DESCRIPTION, SELLPRICE\*0.15 **AS** "NEW\_PRICE"

**FROM** product\_master1 **WHERE** SELLPRICE > 500;

**Input:**

****

**Output:**

Graphical user interface, text, application

Description automatically generated

1. **List the names, city of the clients who are not in ‘Maharashtra.’**

**Input Query:**

**SELECT** **NAME** , city

**FROM** client\_master

**WHERE** **NOT** state ='Maharashtra';

**Input:**

****

**Output:**

A picture containing table

Description automatically generated

1. **Count the total number of orders.**

**Input Query:**

**SELECT** **COUNT**(orderno)

**FROM** sales\_order;

**Input:**

****

**Output:**

A picture containing graphical user interface

Description automatically generated

1. **Calculate the average price of all orders.**

**Input Query:**

**SELECT** **AVG** (costprice)

**FROM** product\_master1;

**Input:**

****

**Output:**

A picture containing shape

Description automatically generated

1. **Determine the maximum and minimum product prices. Rename the output as max\_price and min\_price**.

**Input Query:**

**SELECT** **MAX**(costprice) **AS** "Max\_Price"

**FROM** product\_master1;

**SELECT** **MIN**(costprice) **AS** "Min-price"

**FROM** product\_master1;

**Input:**

****

**Output:**

Table

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

1. **Count the number of products having price less than or equal to 500.**

**Input Query:**

**SELECT** **COUNT**(costprice)

**FROM** product\_master1

**where** costprice<=500;

**Input:**

**A picture containing graphical user interface

Description automatically generated**

**Output:**

Graphical user interface, application, Word

Description automatically generated

1. **List all products whose qtyonhand is less than reorderlvl.**

**Input Query:**

**SELECT** description

**FROM** product\_master1

**WHERE** qtyonhand < reorderlvl;

**Input:**

**Background pattern

Description automatically generated with low confidence**

**Output:**



**VIVA VOCE QUESTIONS**

**Q.1 What are pattern matching operation?**

**Ans:** SQL pattern matching allows you to search for patterns in data if you don't know the exact word or phrase you are seeking. This kind of SQL query uses wild card characters to match a pattern rather than specifying it exactly.

**Q.2 What are different variants of like command?**

**Ans:**

|  |  |
| --- | --- |
| LIKE Operator | DESCRIPTION |
| WHERE CustomerName LIKE ‘a%’ | Finds any value that starts with “a” |
| WHERE CustomerName LIKE ‘%a’ | Finds any value that ends with “a” |
| WHERE CustomerName LIKE ‘%or%’ | Finds any value that has “or” in any position |
| WHERE CustomerName LIKE ‘\_r%’ | Finds any value that has “r” in the second position. |
| WHERE CustomerName LIKE ‘a\_%\_%’ | Finds any value that starts with “a” and are at least 3 characters in length. |
| WHERE ContactName LIKE ‘a%o’ | Finds any value that starts with “a” and ends with “o”. |

**Q.3 What are different Logical operations?**

**Ans:**

|  |  |
| --- | --- |
| **OPERATOR** | **DESCRIPTION** |
| ALL | True if all of the subquery values meet the condition |
| AND | True if all the conditions separated by AND is true. |
| ANY | True if any of the subquery meets the condition. |
| BETWEEN | True if operand is within the range of comparisons. |
| EXISTS | True if the subquery returns one or more records. |
| IN | True if the operand is equal to one of a list of expressions. |
| NOT | Displays a record if the condition(s) is NOT TRUE |
| OR | True if any of the conditions separated by OR is true. |

**Q.4 What is difference between IN and BETWEEN command?**

**Ans:** The IN command allows you to specify multiple values in a WHERE clause. it is a shorthand for multiple OR conditions.

Whereas BETWEEN command select values within a given range the values can be numbers, text or dates. The BETWEEN operator is inclusive: begin an end value are included.