# **Operators:**

SQL has several operators that can be used in queries, including:

**Range operator** (BETWEEN): Used to retrieve data within a specific range of values. For example: WHERE column1 BETWEEN 1 AND 10.

**Membership operator** (IN): Used to retrieve data that matches a list of values. For example: WHERE column1 IN ('value1', 'value2', 'value3').

# **Arithmetic Operators**

+ Add

- Subtract

\* Multiply

/ Divide

Examples

a.) SELECT 30 + 20;

b.) SELECT 30 - 20;

c.) SELECT 30 \* 20;

d.) SELECT 30 / 20;

**Comparison Operators:**

Comparison operators are used to compare two values. The result of a comparison operator is a boolean value (TRUE or FALSE). The commonly used comparison operators in SQL are:

= : Equal to

<> or != : Not equal to

< : Less than

> : Greater than

<= : Less than or equal to

>= : Greater than or equal to

For example, to select all the rows in a table where the value of the column "age" is greater than or equal to 18, you can use the following SQL query:

SELECT \* FROM table1 WHERE age >= 18;

**Logical Operators:**

Logical operators are used to combine multiple conditions using logical operators. The commonly used logical operators in SQL are:

AND : Returns true if both conditions are true

OR : Returns true if either condition is true

NOT : Reverses the logical state of a condition

For example, to select all the rows in a table where the value of the column "age" is greater than or equal to 18 and the value of the column "gender" is "male", you can use the following SQL query:

SELECT \* FROM table1 WHERE age >= 18 AND gender = 'male';

# 

# DISTINCT and LIMIT:

**DISTINCT** is used to retrieve unique values from a column. For example: SELECT DISTINCT column1. This can be useful for identifying unique values in a large dataset.

**LIMIT** is used to limit the number of rows returned by a query. For example: SELECT column1 LIMIT 10. This is useful when you only want to see a small sample of the data.

# Wildcards:

Wildcards are special characters that can be used in SQL to match one or more characters in a string. Here are the two commonly used wildcards in SQL:

**Percent sign (%):**

The percent sign (%) is used to match any sequence of zero or more characters. It can be used to search for strings that contain a specific sequence of characters in any position. For example, to search for all names that begin with "Joh" in a table, you can use the following SQL query:

SELECT \* FROM table1 WHERE name LIKE 'Joh%';

This will match all the names that start with "Joh", such as "John", "Johanna", "Johannes", etc.

**Underscore ():**

The underscore () is used to match any single character. It can be used to search for strings that have a specific character in a specific position. For example, to search for all names that have a fourth letter of "n" in a table, you can use the following SQL query:

SELECT \* FROM table1 WHERE name LIKE '\_\_\_n%';

This will match all the names that have "n" as their fourth letter, such as "Benjamin", "Sandra", "Franklin", etc.

You can also combine the two wildcards to create more complex searches. For example, to search for all names that contain "an" in the third and fourth position in a table, you can use the following SQL query:

SELECT \* FROM table1 WHERE name LIKE '\_\_an%';

This will match all the names that have "an" as their third and fourth letters, such as "Sandra", "Francine", "Vance", etc.

In summary, wildcards can be used to search for patterns in strings, making it easier to find specific data in a table.

# String functions:

string functions are used to manipulate string data. Here are some commonly used string functions in SQL:

**CONCAT():**

The CONCAT() function is used to concatenate two or more strings into one. For example, to concatenate the first name and last name columns in a table, you can use the following SQL query:

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name FROM table1;

This will return a new column called full\_name that contains the concatenated values of the first\_name and last\_name columns.

**SUBSTRING():**

The SUBSTRING() function is used to extract a portion of a string. It takes three arguments: the string to be extracted, the starting position of the substring, and the length of the substring. For example, to extract the first three characters of the first\_name column in a table, you can use the following SQL query:

SELECT SUBSTRING(first\_name, 1, 3) AS first\_three FROM table1;

This will return a new column called first\_three that contains the first three characters of the first\_name column.

**UPPER() and LOWER():**

The UPPER() function is used to convert a string to uppercase, while the LOWER() function is used to convert a string to lowercase. For example, to convert the last\_name column in a table to uppercase, you can use the following SQL query:

SELECT UPPER(last\_name) AS last\_name\_upper FROM table1;

This will return a new column called last\_name\_upper that contains the last\_name column in uppercase.

**LENGTH():**

The LENGTH() function is used to return the length of a string. For example, to find the length of the first\_name column in a table, you can use the following SQL query:

SELECT LENGTH(first\_name) AS name\_length FROM table1;

This will return a new column called name\_length that contains the length of the first\_name column.

**REPLACE():**

The REPLACE() function is used to replace a portion of a string with a new string. It takes three arguments: the string to be replaced, the string to replace it with, and the string to search for. For example, to replace all occurrences of the string "Mr." with the string "Sir" in the title column of a table, you can use the following SQL query:

SELECT REPLACE(title, 'Mr.', 'Sir') AS new\_title FROM table1;

This will return a new column called new\_title that contains the title column with all occurrences of "Mr." replaced with "Sir".

These are just a few examples of the string functions available in SQL. Other functions include TRIM(), LTRIM(), and RTRIM(), among others.

# Numeric functions:

numeric functions are used to perform mathematical operations on numeric data. Here are some commonly used numeric functions in SQL:

**SUM():**

The SUM() function is used to calculate the sum of a set of values. For example, to find the total sales for a particular product in a table, you can use the following SQL query:

SELECT SUM(sales) AS total\_sales FROM table1 WHERE product\_id = '123';

This will return a new column called total\_sales that contains the sum of the sales column for all rows where the product\_id is '123'.

**AVG():**

The AVG() function is used to calculate the average value of a set of values. For example, to find the average salary for employees in a table, you can use the following SQL query:

SELECT AVG(salary) AS avg\_salary FROM table1;

This will return a new column called avg\_salary that contains the average value of the salary column.

**MAX() and MIN():**

The MAX() function is used to find the maximum value in a set of values, while the MIN() function is used to find the minimum value. For example, to find the highest and lowest temperatures recorded in a table, you can use the following SQL query:

SELECT MAX(temperature) AS highest\_temp, MIN(temperature) AS lowest\_temp FROM table1;

This will return a new table with two columns: highest\_temp, which contains the maximum value of the temperature column, and lowest\_temp, which contains the minimum value.

**ROUND():**

The ROUND() function is used to round a number to a specified number of decimal places. For example, to round the salary column in a table to two decimal places, you can use the following SQL query:

SELECT ROUND(salary, 2) AS rounded\_salary FROM table1;

This will return a new column called rounded\_salary that contains the salary column rounded to two decimal places.

**ABS():**

The ABS() function is used to find the absolute value of a number. For example, to find the absolute difference between the sales and expenses columns in a table, you can use the following SQL query:

SELECT ABS(sales - expenses) AS net\_income FROM table1;

This will return a new column called net\_income that contains the absolute difference between the sales and expenses columns.

These are just a few examples of the numeric functions available in SQL. Other functions include CEILING(), FLOOR(), POWER(), SQRT(), among others.

# Temporal functions:

Temporal functions are used to manipulate date and time data. Here are some commonly used temporal functions in SQL

**DATE:**

The DATE function is used to extract the date portion of a date/time value. The syntax is as follows:

DATE(datetime\_expression)

For example, if you have a timestamp column called "created\_at", you can use the DATE function to retrieve the date portion of the timestamp like this:

SELECT DATE(created\_at) as date\_only

FROM table1

This will return a table with a single column called "date\_only", which contains the date portion of each timestamp in the "created\_at" column.

**YEAR:**

The YEAR function is used to extract the year portion of a date/time value. The syntax is as follows:

YEAR(datetime\_expression)

For example, if you have a timestamp column called "created\_at", you can use the YEAR function to retrieve the year portion of the timestamp like this:

SELECT YEAR(created\_at) as year\_only

FROM table1

This will return a table with a single column called "year\_only", which contains the year portion of each timestamp in the "created\_at" column.

**MONTH:**

The MONTH function is used to extract the month portion of a date/time value. The syntax is as follows:

MONTH(datetime\_expression)

For example, if you have a timestamp column called "created\_at", you can use the MONTH function to retrieve the month portion of the timestamp like this:

SELECT MONTH(created\_at) as month\_only

FROM table1

This will return a table with a single column called "month\_only", which contains the month portion of each timestamp in the "created\_at" column.

**DAY:**

The DAY function is used to extract the day portion of a date/time value. The syntax is as follows:

DAY(datetime\_expression)

For example, if you have a timestamp column called "created\_at", you can use the DAY function to retrieve the day portion of the timestamp like this:

SELECT DAY(created\_at) as day\_only

FROM table1

This will return a table with a single column called "day\_only", which contains the day portion of each timestamp in the "created\_at" column.

**NOW:**

The NOW function is used to retrieve the current date and time. The syntax is as follows:

NOW()

For example, if you want to retrieve the current date and time in a query, you can use the NOW function like this:

SELECT NOW() as current\_time

This will return a table with a single column called "current\_time", which contains the current date and time.

These are just a few examples of the temporal functions available in SQL. Other functions include HOUR, MINUTE, SECOND, and TIMESTAMPDIFF, which can be used to calculate the difference between two timestamps.