CLAUSES

WHERE : A WHERE clause is an optional part of a select Expression, DELETE statement, or UPDATE statement. The WHERE clause lets you select rows based on a Boolean expression.

Syntax :

SELECT * FROM Table_Name WHERE Condition ;

Arithmetic Operators

Description	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/

Comparison Operators

Description	Operator
Less Then	<
Less Then or Equal To	<=
Greater Then	>
Greater Then or Equal To	>=
Equal To	=
Not Equal to	**!= or <>**

SQL AND, OR and NOT Operators

The **AND** and **OR** operators are used to filter records based on more than one condition:

- The **AND** operator displays a record if all the conditions separated by **AND** are TRUE.

  Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

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

  Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

- The **NOT** operator displays a record if the condition(s) is **NOT** TRUE.

  Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Aggregate functions in SQL

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Various Aggregate Functions:

1) **Count()**
2) **Sum()**
3) **Avg()**
4) **Min()**
5) **Max()**

WORKER

Id	Name	Salary
1	A	10000
2	B	20000
3	C	30000
4	D	40000
5	E	50000
6	F	34000

Count():

Count(*): Returns total number of records.
Count(salary): Return number of Non-Null values over the column salary.
Count(Distinct Salary): Return number of distinct Non-Null values over the column salary.

Sum():

sum(salary): Sum all Non Null values of Column salary.
sum(Distinct salary): Sum of all distinct Non-Null values.

Avg():

Avg(salary) = Sum(salary) / count(salary) = 310/5
Avg(Distinct salary) = sum(Distinct salary) / Count(Distinct Salary) = 250/4

Min()/Max():

Min(salary): Minimum value in the salary column except NULL.
Max(salary): Maximum value in the salary.

String Operation

LIKE

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
- There are two wildcards often used in conjunction with the LIKE operator:
  - The percent sign (%) represents zero, one, or multiple characters
  - The underscore sign (_) represents one, single character

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

LIKE Operator	Description
LIKE 'a%'	Finds any values that start with "a"
LIKE '%a'	Finds any values that end with "a"
LIKE '%or%'	Finds any values that have "or" in any position
LIKE '_r%'	Finds any values that have "r" in the second position
LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

CUSTOMER TABLE

C_ID	C_Name	Address	City	PostalCode	Country
1	GOVIND RATHOD	HINJEWADI PHASE-II	PUNE	12209	Germany
2	SATYAM SINGH	KASBA	BARAMATI	05021	INDIA
3	DATTA JADAV	MIRAROAD	MUMBAI	05023	Mexico
4	SAURAV VARMA	KHARADI	PUNE	411014	UK
5	SANKALP SHARMA	BOROVALI	MUMBAI	201201	Mexico
6	SAMEER BHAT	RAJENDRA NAGAR	DELHI	68306	INDIA
7	DEEPAK KOKRE	BOROVALI	MUMBAI	67000	Germany

- **selects all customers with a Customer Name starting with “S":**

```
SELECT * FROM Customers
WHERE C_Name LIKE ’S%';
```

- **selects all customers with a Customer City ending with “i":**

```
SELECT * FROM Customers
WHERE City LIKE ’%i’;
```

- **selects all customers with a Customer Name that have “EE" in any position:**

```
SELECT * FROM Customers
WHERE C_Name LIKE ’%EE%';
```

- **selects all customers with a Customer Name that have “A" in the second position:**

```
SELECT * FROM Customers
WHERE C_Name LIKE ’_A%';
```

- **selects all customers with a Customer Name that starts with "a" and are at least 3 characters in length:**

```
SELECT * FROM Customers
WHERE C_Name LIKE 'a__%';
```

- **selects all customers with a Address that starts with “k" and ends with “i":**

```
SELECT * FROM Customers
WHERE  Address  LIKE 'a%o’;
```

- **selects all customers with a C_Name that does NOT start with "a":**

```
SELECT * FROM Customers
WHERE C_Name NOT LIKE 'a%';
```

DISTINCT

- The SELECT DISTINCT statement is used to return only distinct (different) values.

- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

Syntax:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.
The IN operator is a shorthand for multiple OR conditions

Syntex:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
The BETWEEN operator is inclusive: begin and end values are included.

Syntex:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

GROUP BY

- SQL GROUP BY statement is used to arrange identical data into groups. The GROUP BY statement is used with the SQL SELECT statement.
- The GROUP BY statement follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
- The GROUP BY statement is used with aggregation(min, max, count, sum, avg) function.

Syntax

SELECT column FROM table_name WHERE conditions
GROUP BY column

PRODUCT	COMPANY	QTY	RATE	COST
Item1	COMPANY-A	2	10	20
Item2	COMPANY-B	3	25	75
Item3	COMPANY-C	2	30	60
Item4	COMPANY-B	5	10	50
Item5	COMPANY-A	2	20	40
Item6	COMPANY-C	3	25	75
Item7	COMPANY-A	5	30	150

EXAMPLE:

SELECT COMPANY, COUNT(*) FROM PRODUCT GROUP BY COMPANY;

HAVING

- HAVING clause is used to specify a search condition for a group or an aggregate.
- Having is used in a GROUP BY clause. If you are not using GROUP BY clause then you can use HAVING function like a WHERE clause.

Syntax:

SELECT *column1, column2 FROM table_name WHERE conditions*
GROUP BY column1, column2 HAVING conditions ;
SELECT *column1, column2 FROM table_name WHERE conditions*
HAVING conditions ;

EXAMPLE:

SELECT COMPANY, COUNT(*) FROM PRODUCT GROUP BY COMPANY
HAVING COUNT(*)>2;

ORDER BY

- The ORDER BY clause sorts the result-set in ascending or descending order.
- It sorts the records in ascending order by default. DESC keyword is used to sort the records in descending order.

Syntax:

```
SELECT column1, column2  FROM table_name  WHERE condition
ORDER BY column1, column2... ASC|DESC;
```

ASC: It is used to sort the result set in ascending order by expression.

DESC: It sorts the result set in descending order by expression.

CUSTOMER

CUSTOMER_ID	NAME	ADDRESS
12	Kathrin	US
23	David	Bangkok
34	Alina	Dubai
45	John	UK
56	Harry	US

1. ***SELECT * FROM CUSTOMER ORDER BY NAME;***
2. ***SELECT * FROM CUSTOMER ORDER BY NAME DESC;***

Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of that query.
- An alias is created with the AS keyword.

Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

Table Syntax

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

-- THE END --