SQL stands for Structured Query Language.

# SELECT:

It is used to retrieve data from database.

SELECT name

FROM students;

# FROM:

It is used to mention the table from where we want to get the columns will all rows.

SELECT age

FROM students;

# TOP (5):

Top is used to limit the first rows till how much we want to retrieve data from rows. The parameter 5 means that we need top 5 rows from the column mentioned.

SELECT TOP (5) name

FROM students;

# PERCENT:

If we mention the percent keyword after top so instead of 5 rows, it’ll return 5% rows from top.

SELECT TOP (10) PERCENT name

FROM students;

# SELECT \*:

Using \* with select means that return all columns and rows from the required table.

SELECT \*

FROM students;

# AS:

It’s used for aliasing. Used to give the selected column a temporary alias (alternative name) for the output. The alias doesn’t change the column name in the database it’s just for readability in the result.

SELECT name AS student\_name

FROM students;

# DISTINCT:

The DISTINCT keyword is applied after SELECT. It filters out duplicate rows based on the columns listed in the SELECT statement.

SELECT DISTINCT age

FROM students;

# Order By:

Queries return set of data. And its not necessary that the data which we’ll get will be in the same sequence always. That’s why we use ORDER keyword to get the data in a specific order always.

The given below code will order in ascending. You want in descending then you can add DESC keyword after column name

SELECT TOP(10) PERCENT name

FROM students

Order By name, id

Specifying multiple columns in an ORDER BY clause allows you to define a more detailed sorting order. The database will sort the results **first** by the first column, and then **within rows that have the same value for the first column**, it will sort by the second column.

SELECT TOP(10) PERCENT name

FROM students

Order By name DESC, id

This ordering can be applied to columns having text as well.

# Defining criteria

We can define criteria with some conditions as well to get data from database.

Select name

From students

Where age > 15;

Or Where date = ’12-2-2023’

# <>:

In sql <> means those values those are either bigger or smaller than mentioned number and not equal to.

Where total BETWEEN 10 AND 30:

In this query between and “and” are used collectively to select data between a specific range.

If we want to exclude that range from our required data than we can use

Where total NOT BETWEEN 10 AND 30

# Selecting and excluding NULL values:

For selecting those values from column where the value is null we can use this code

Select name

From students

Where name IS NULL;

# And if we want to exclude null data then:

Select name

From students

Where name IS NOT NULL;

Several Condtions:

We can apply several conditions using AND between condtions.

Select name

From students

Where age > 15

AND gender = ‘Male’;

We can also use OR between them.

Select name

From students

Where age > 15

OR gender = ‘Male’;

IN:

The IN keyword in SQL is used to filter records where a column's value matches any value in a specified list. It acts as a shorthand for multiple OR conditions.

SELECT \* FROM employees

WHERE department IN ('HR', 'Finance', 'IT');

Returns all employees who work in the HR, Finance, or IT departments.

LIKE:

The LIKE keyword in SQL is used in a WHERE clause to search for a specified pattern in a column. It is commonly used with wildcard characters:

**Wildcards**:

* %: Represents zero, one, or multiple characters.
* \_: Represents a single character.

SELECT \* FROM employees

WHERE name LIKE 'J%';

Returns all employees whose names start with "J".

SELECT \* FROM employees

WHERE name LIKE 'A\_n';

Returns all employees whose names have "A" as the first letter, followed by any one character, and end with "n" (e.g., "Ann").

SELECT \* FROM employees

WHERE name LIKE '%son%';

Returns all employees whose names contain "son" anywhere.

Aggregations in SQL

SUM:

The SUM function in SQL calculates the total (sum) of a numeric column. It is often used to perform aggregations in combination with GROUP BY or on its own.

SELECT SUM(salary) AS total\_salary

FROM employees;

This query calculates the total salary of all employees.

But if we want to sum several columns so its necessary to add several sum keywords otherwise we’ll get an error. And it necessary to add AS and a temporary name because when the sum is returned there will be no column name with it.

COUNT:

The COUNT function in SQL is used to count the number of rows in a table or the number of non-NULL values in a column.

SELECT COUNT(\*) AS total\_rows

FROM employees;

This query counts the total number of rows in the employees table, including rows with NULL values.

SELECT COUNT(salary) AS total\_salaries

FROM employees;

This query counts the number of non-NULL values in the salary column.

If we want to get unique values in result then:

SELECT COUNT(DISTINCT department) AS unique\_departments

FROM employees;

This query counts the number of unique departments in the department column.

MIN or MAX:

The MIN function in SQL is used to return the smallest (minimum) value in a column. It is commonly used to find the lowest value in a dataset, such as the lowest salary or earliest date.

SELECT MIN(salary) AS minimum\_salary

FROM employees;

**AVG Function in SQL**

The AVG function in SQL calculates the average (mean) of a numeric column. It is useful for finding average values, such as average salaries, prices, or scores in a dataset.

SELECT AVG(salary) AS average\_salary

FROM employees;

This query calculates the average salary of all employees.

LEN:

The LEN function in SQL is used to calculate the **length** of a string (the number of characters in a given text). It is often used to analyze text data, such as finding the length of names, codes, or other string-based columns.

SELECT LEN(name) AS name\_length

FROM employees;

**LEFT and RIGHT Functions in SQL**

The LEFT and RIGHT functions are used to extract a specified number of characters from the beginning or the end of a string, respectively. These functions are helpful in text manipulation, such as extracting prefixes, suffixes, or parts of a string.

SELECT LEFT(name, 3) AS first\_three\_characters

FROM employees;

CHARINDEX:

The CHARINDEX function in SQL is used to find the starting position of a substring within a string. It is case-insensitive in most SQL implementations and is commonly used to locate specific characters or words in a text column.

SELECT CHARINDEX('SQL', 'Learn SQL with ease') AS position;

**SUBSTRING Function**

The SUBSTRING function extracts a portion of a string, starting from a specified position and for a specified length.

SELECT SUBSTRING('Hello World', 7, 5) AS extracted\_string;

(Extracts 5 characters starting from the 7th position.)

**REPLACE Function**

The REPLACE function is used to replace all occurrences of a specified substring within a string with another substring.

SELECT REPLACE('Hello World', 'World', 'SQL') AS replaced\_string;

(Replaces "World" with "SQL".)

GROUP BY

The **GROUP BY** clause in SQL is used to group rows that have the same values in specified columns into aggregate data. It is often used with aggregate functions like SUM, COUNT, AVG, MIN, and MAX to perform operations on each group of rows.

SELECT department, COUNT(\*) AS employee\_count

FROM employees

GROUP BY department;

**HAVING Clause in SQL**

The **HAVING** clause in SQL is used to filter groups of data that are created using the **GROUP BY** clause. It allows you to apply conditions to aggregated data (e.g., results of SUM, COUNT, AVG, etc.).

**Key Points**

1. **Used with GROUP BY**:
   * HAVING is applied to groups of rows after the aggregation.
   * Similar to the WHERE clause, but it works on aggregate data.
2. **Difference Between WHERE and HAVING**:
   * **WHERE**: Filters rows **before** grouping and aggregation.
   * **HAVING**: Filters groups **after** aggregation.

SELECT region, SUM(sales\_amount) AS total\_sales

FROM sales

GROUP BY region

HAVING SUM(sales\_amount) > 3000;