**SQL**

* SQL stands for **Structured Query Language**

**PURPOSE OF SQL**

* SQL can **execute queries** against a database
* SQL can **retrieve data** from a database
* SQL can **insert, delete, update records** in a database
* SQL can **create new databases**
* SQL can **create new tables, views** in a database

**SQL CAN BE USED IN A WEBSITE**

To build a **web site that shows data from a database**, we need:

* An **RDBMS database program** (i.e. MS Access, SQL Server, MySQL)
* A **server-side scripting language**, like PHP or ASP
* **SQL** to get the data you want
* **HTML / CSS** to style the page

**RDBMS**

* RDBMS stands for **Relational Database Management System**.
* RDBMS is the **basis for SQL**, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
* The data in RDBMS is stored in database objects called **tables**.
* A table is a collection of **related data entries** and it consists of **columns (fields)** and **rows(records)**.

**SQL keywords are NOT case sensitive**

**Semicolon** is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

Some of the SQL Commands :-

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table

**SQL Constraints**

* **Constraints** are the **rules enforced** on **data columns** on a table. These are used to **limit the type of data** that can go into a table.
* This ensures the **accuracy and reliability** of the data in the database.
* Some of the most commonly used constraints available in SQL:

**NOT NULL Constraint**: Ensures that a column cannot have a NULL value. **DEFAULT Constraint:** Provides a default value for a column when none is specified.

**UNIQUE Constraint**: Ensures that all the values in a column are different. **PRIMARY Key:** Uniquely identifies each row/record in a database table. • **FOREIGN Key:** Uniquely identifies a row/record in any another database table.

**CHECK Constraint**: The CHECK constraint ensures that all values in a column satisfy certain conditions.

**INDEX**: Used to create and retrieve data from the database very quickly.

## CREATE TABLE Statement

The **CREATE TABLE** statement is used to **create a new table** in a database.

### **Syntax:-**

CREATE TABLE table\_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
   ....);

## DROP TABLE Statement

The **DROP TABLE** statement is used to **drop an existing table** in a database.

### **Syntax:-**

DROP TABLE table\_name;

## TRUNCATE TABLE

The **TRUNCATE TABLE** statement is used to **delete the data inside a table**, but **not the table itself**.

### **Syntax:-**

TRUNCATE TABLE table\_name;

## SELECT Statement

The **SELECT**statement is used to **select data** from a database.

### **Syntax :-**

* SELECT column1, column2, ...FROM table\_name;

//Here only column1, column2… would get displayed.

* SELECT \* FROM table\_name;

//Here all the contents of the table ( \* ) would get displayed.

## SELECT DISTINCT Statement

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

### **Syntax :-**

SELECT DISTINCT column1, column2, ...FROM table\_name;

**The main difference between Unique and Distinct in SQL** is that**Unique helps to ensure that all the values in a column are different while Distinct helps to remove all the duplicate records when retrieving the records from a table.**

## WHERE Clause

The **WHERE** clause is used to **filter records**. It is used to extract only those records that **fulfil a specified condition**. It is used with **UPDATE and DELETE** statements too.

**Syntax :-**

SELECT column1, column2, ...FROM table\_name WHERE condition;

## AND, OR and NOT Operators

The **WHERE** clause can be combined with **AND, OR, and NOT** operators.

The **AND and OR operators** are used to filter records based on **more than one condition.** They display records if the conditions separated by AND and OR are **TRUE**.

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

### **AND Syntax :-**

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

### **OR Syntax :-**

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

### **NOT Syntax :-**

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

## ORDER BY Keyword

The **ORDER BY** keyword is used to sort the result-set in **ascending or descending order**.

**By default** its orders them in **ascending order**. To sort the records in descending order, use the **DESC keyword**.

### **Syntax:-**

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

## INSERT INTO Statement

The **INSERT INTO** statement is used to **insert new records** in a table.

### **Syntax :-**

It is possible to write the INSERT INTO statement in two ways:

**1.** **To insert values in specific columns :-**

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

**2.** **To add values for all the columns of the table :-**

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

**NULL VALUE :-**

A **NULL value** in a table is a value in a field that appears to be **blank (no value)**

A **field with a NULL value** is the one that has been **left blank** during a **record creation**.

### **IS NULL Syntax :-**

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

### **IS NOT NULL Syntax :-**

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NOT NULL;

## UPDATE Statement

The **UPDATE statement** is used to **modify the existing records** in a table.

### **Syntax :-**

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

The **WHERE** clause specifies which **record(s)** should be updated. If there is no **WHERE** clause, then **all records** in the table will get updated.

## DELETE Statement

The **DELETE statement** is used to **delete existing records** in a table.

### **Syntax :-**

DELETE FROM table\_name WHERE condition;

The **WHERE** clause specifies which **record(s)** should be updated. If there is no **WHERE** clause, then **all records** in the table will get updated.

**For deleting all records** :-

DELETE FROM table\_name ;

**LIKE Operator**

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;

|  |  |
| --- | --- |
| WHERE *column\_name* LIKE 'a%' | Finds any values that start with "a" |
| WHERE *column\_name* LIKE '%a' | Finds any values that end with "a" |
| WHERE *column\_name* LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE *column\_name* LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE *column\_name* LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |
| WHERE *column\_name* LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE *column\_name* LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

## 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.**

### **Syntax:-**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

**or:**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (*SELECT* STATEMENT);

## 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**.

### **Syntax:-**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

## GROUP BY Statement

The **GROUP BY** statement **groups rows** that have the **same values** into summary rows.The **GROUP BY** statement is often used with aggregate functions **(COUNT(), MAX(), MIN(), SUM(), AVG())** to group the result-set by one or more columns.

### **Syntax:-**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

## HAVING Clause

The **HAVING clause** is used with **aggregate functions**.**WHERE keyword** **cannot** be used with **aggregate functions**.

### **Syntax :-**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

## ALTER TABLE Statement

The **ALTER TABLE** statement is used to **add, delete, or modify columns** in an existing table.It is also used to add and drop various constraints on an existing table.

## ALTER TABLE - ADD Column

To add a column in a table

### **Syntax :-**

ALTER TABLE table\_name  
ADD column\_name datatype;

## ALTER TABLE - DROP COLUMN

To delete a column in a table

### **Syntax :-**

ALTER TABLE table\_name  
DROP COLUMN column\_name;

## ALTER TABLE - MODIFY COLUMN

To change the data type of a column in a table.

### **Syntax :-**

ALTER TABLE table\_name  
MODIFY COLUMN column\_name datatype;

## JOIN

A **JOIN** clause is used to **combine rows from two or more tables, based on a related column between them.**

**Different Types of SQL JOINs**

Here are the different types of the **JOINs** in SQL:

* **(INNER) JOIN**: Returns records that have matching values in both tables
* **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
* **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
* **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

### **INNER JOIN Syntax :-**

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name = table2.column\_name;

### **LEFT JOIN Syntax:-**

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

### **RIGHT JOIN Syntax:-**

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

### **FULL OUTER JOIN Syntax:-**

SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2ON table1.column\_name = table2.column\_nameWHERE condition;

**Difference Between Unique And Distinct**

* The main difference between **unique** and **distinct** is that **UNIQUE is a constraint** that is **used on the input of data and ensures data integrity**. While **DISTINCT** keyword is used when **we want to query our results or in other words, output the data**.
* Both UNIQUE and DISTINCT key words ensure the same thing, i.e. **data is not repeated**.
* **One makes sure during input and other during the output.**