# SQL

## SELECT

Reads the data from the SQL database and shows it as the output to the database user.

SELECT column\_name1, column\_name2, .…, column\_nameN

[ FROM table\_name ]

[ WHERE condition ]

[ ORDER BY order\_column\_name1 [ ASC | DESC ], .... ];

**Example of SELECT Statement:**

SELECT Emp\_ID, First\_Name, Last\_Name, Salary, City

FROM Employee\_details

WHERE Salary = 100000

ORDER BY Last\_Name DESC

## UPDATE

Modified stored data in db

**Syntax of UPDATE Statement:**

**UPDATE** table\_name

**SET** column\_name1 = new\_value\_1, column\_name2 = new\_value\_2, ...., column\_nameN = new\_value\_N

[ **WHERE** CONDITION ];

**Example of UPDATE Statement:**

UPDATE Employee\_details

SET Salary = 100000

WHERE Emp\_ID = 10;

## DELETE

Deletes the stored record from table

**Syntax of DELETE Statement:**

**DELETE** **FROM** table\_name

[ **WHERE** CONDITION ];

**Example of DELETE Statement:**

DELETE FROM Employee\_details

WHERE First\_Name = 'Sumit';

## CREATE

This SQL statement creates the new table in the SQL database.

**Syntax of CREATE TABLE Statement:**

1. **CREATE** **TABLE** table\_name

(

column\_name1 data\_type [column1 **constraint**(s)],

column\_name2 data\_type [column2 **constraint**(s)],

.....

.....,

column\_nameN data\_type [columnN **constraint**(s)],

**PRIMARY** **KEY**(one or more col)

);

**Example of CREATE TABLE Statement:**

CREATE TABLE Employee\_details(

Emp\_Id NUMBER(4) NOT NULL,

First\_name VARCHAR(30),

Last\_name VARCHAR(30),

Salary Money,

City VARCHAR(30),

PRIMARY KEY (Emp\_Id)

);

## ALTER

SQL statement adds, deletes, and modifies the columns of the table in the SQL database.

**Syntax of ALTER TABLE Statement:**

**ALTER** **TABLE** table\_name **ADD** column\_name datatype[(**size**)];

The above SQL alter statement adds the column with its datatype in the existing database table.

Modify col datatype

ALTER TABLE table\_name

MODIFY COLUMN column\_name data\_type;

ALTER TABLE employee\_details

RENAME COLUMN emp\_id TO eid;

The above 'SQL alter statement' renames the old column name to the new column name of the existing database table.

**ALTER** **TABLE** table\_name **DROP** **COLUMN** column\_name;

ADVERTISEMENT

The above SQL alter statement deletes the column of the existing database table.

**Example of ALTER TABLE Statement:**

ALTER TABLE Employee\_details

**ADD** Designation **VARCHAR**(18);

## DROP TABLE Statement

This SQL statement deletes or removes the table and the structure, views, permissions, and triggers associated with that table.

**Syntax of DROP TABLE Statement:**

**DROP** **TABLE tablename** [ IF EXISTS ]

table\_name1, table\_name2, ……, table\_nameN;

DROP TABLE Employee\_details;

## DROP DATABASE Statement

This SQL statement deletes the existing database with all the data tables and views from the database management system.

**Syntax of DROP DATABASE Statement:**

**DROP** **DATABASE** database\_name;

**Example of DROP DATABASE Statement:**

DROP DATABASE Company;

## INSERT INTO Statement

This SQL statement inserts the data or records in the existing table of the SQL database. This statement can easily insert single and multiple records in a single query statement.

**Syntax of insert a single record:**

**INSERT** **INTO** table\_name

(

column\_name1,

column\_name2, .…,

column\_nameN

)

**VALUES**

(value\_1,

value\_2, ..…,

value\_N

);

**Example of insert a single record:**

INSERT INTO Employee\_details

(

Emp\_ID,

First\_name,

Last\_name,

Salary,

City

)

VALUES

(101,

Akhil,

Sharma,

40000,

Bangalore

), (101,

Akhil,

Sharma,

40000,

Bangalore

);

## TRUNCATE TABLE

SQL statement deletes all the stored records from the table of the SQL database.

**Syntax of TRUNCATE TABLE Statement:**

**TRUNCATE** **TABLE** table\_name;

Eg: Truncate table employee\_details;

## DESCRIBE

This SQL statement tells something about the specified table or view in the query.

**Syntax of DESCRIBE Statement:**

DESCRIBE table\_name | view\_name;

## DISTINCT

Eliminates all duplicate records from the result returned by the SQL query

**Syntax of DISTINCT Clause:**

**SELECT** **DISTINCT** column\_name1, column\_name2, ...

**FROM** table\_name;

**Example of DISTINCT Clause:**

SELECT DISTINCT City, Salary

FROM Employee\_details;

## CREATE INDEX

The **CREATE INDEX** command is used to **create indexes** in **tables** (allows duplicate values).

**Syntax of CREATE INDEX Statement:**

**CREATE** **INDEX** index\_name

**ON** table\_name ( column\_name1, column\_name2, …, column\_nameN );

**Example of CREATE INDEX Statement:**

CREATE INDEX idx\_First\_Name

ON employee\_details (First\_Name);

## DROP INDEX

This SQL statement deletes the existing index of the SQL database table.

**Syntax of DROP INDEX Statement:**

**DROP** **INDEX** index\_name;

**Example of DROP INDEX Statement:**

DROP INDEX idx\_First\_Name;

## USE Statement

This SQL statement selects the existing SQL database. Before performing the operations on the database table, you have to select the database from the multiple existing databases.

**Syntax of USE Statement:**

USE database\_name;

## SQL DATATYPES

Data types are used to represent the nature of the data that can be stored in the database table.

Varchar

Number

Int

Date

Timestamp

<https://www.javatpoint.com/sql-data-types> - for diff sql there are diff types of datatype

## SQL Operators

The manipulation and retrieving of the data are performed with the help of reserved words and characters, which are used to perform arithmetic operations, logical operations, comparison operations, compound operations, etc.

## SQL Arithmetic Operators

* SQL Addition Operator (+)
* SQL Subtraction Operator (-)
* SQL Multiplication Operator (+)
* SQL Division Operator (-)
* SQL Modulus Operator (+)

SQL Comparison Operators

* SQL Equal Operator (=)
* SQL Not Equal Operator (!=)
* SQL Greater Than Operator (>)
* SQL Greater Than Equals to Operator (>=)
* SQL Less Than Operator (<)\
* SQL Less Than Equals to Operator (<=)

## SQL Logical Operators

* SQL ALL operator
* SQL AND operator
* SQL OR operator
* SQL BETWEEN operator
* SQL IN operator
* SQL NOT operator
* SQL ANY operator
* SQL LIKE operator

SQL Set Operators

* SQL Union Operator
* SQL Union ALL Operator
* SQL Intersect Operator
* SQL Minus Operator

## SQL Unary Operators

* SQL Unary Positive Operator
* SQL Unary Negative Operator
* SQL Unary Bitwise NOT Operator

## SQL Bitwise Operators

* Bitwise AND (&)
* Bitwise OR(|)

## **SQL Database**

### **CREATE database**

**CREATE** **DATABASE** Database\_Name;

**CREATE** OR REPLACE **DATABASE** Employee;

SHOW **DATABASE;**

### **DROP Database**

**DROP** **DATABASE** Database\_Name;

### **RENAME Database**

**ALTER** **DATABASE** old\_database\_name **MODIFY** **NAME** = new\_database\_name; **EXEC** sp\_renamedb'old\_database\_name' , 'new\_database\_name'

INMySQL - RENAME **DATABASE** old\_database\_name **TO** new\_database\_name;

### SELECT Database

USE Database\_name

## **SQL TABLE**

### Create table

**CREATE** **TABLE** STUDENTS (

ID **INT** NOT NULL,

**NAME** **VARCHAR** (20) NOT NULL,

AGE **INT** NOT NULL,

ADDRESS **CHAR** (25),

**PRIMARY** **KEY** (ID)

);

Create table using other table

**CREATE** **TABLE** table\_name **AS**

**SELECT** column1, column2,...

**FROM** old\_table\_name **WHERE** ..... ;

The following SQL creates a copy **of** the employee **table**.

**CREATE** **TABLE** EmployeeCopy **AS**

**SELECT** EmployeeID, FirstName, Email

**FROM** Employee;

### Drop Table

It will delete a table

**DROP** **TABLE** "table\_name";

## Having

select max(eid) as minimum from employee\_details where max(eid)>120

aggregate functions are not allowed in WHERE

select max(eid) as minimum from employee\_details having max(eid)>120

Order to use

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING
6. ORDER BY

## CASE

The SQL CASE statement

The CASE statement is SQL's way of handling if/then logic. The CASE statement is followed by at least one pair of WHEN and THEN statements. Because of this pairing, you might be tempted to call this SQL CASE WHEN, but CASE is the accepted term.

Every CASE statement must end with the END statement. The ELSE statement is optional and provides a way to capture values not specified in the WHEN/THEN statements.

SELECT player\_name, weight,

CASE WHEN weight > 250 THEN 'over 250'

WHEN weight > 200 AND weight <= 250 THEN '201-250'

WHEN weight > 175 AND weight <= 200 THEN '176-200'

ELSE '175 or under' END AS weight\_group

FROM benn.college\_football\_players