***SQL- The Programming Language***

SQL is the standard language for dealing with Relational Databases.

SQL is used to insert, search, update, and delete database records.

SQL keywords are NOT case sensitive: select is the same as SELECT

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.

* SELECT - extracts data from a database

COLUMN1, COLUMN2- for specific columns

\*- to select all columns

* SELECT DISTINCT- Return only distinct values
* WHERE- Filter records, not used in only SELECT, but also in UPDATE< DELETE, etc.

= Equal

>Greater than

<Less than

>= Greater than or equal

<= Less than or equal

!= <> Not equal

& Bitwise AND

| Bitwise OR

^ Bitwise exclusive OR

BETWEEN Between a certain Range

LIKE Search for a pattern

e.g- SELECT \* FROM Customers

WHERE City LIKE 's%';

s% means anything after s of as ever no. e.g- Satyabhama

IN To specify multiple values for a column

* 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
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index
* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.
* The NOT operator displays a record if the condition(s) is NOT TRUE.
* The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
* We will have to use the IS NULL and IS NOT NULL operators instead. It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

SELECT column\_names  
FROM table\_name  
WHERE column\_name IS NULL;

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

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

Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

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

DELETE FROM table\_name WHERE condition;

* The LIMIT clause is used to specify the number of records to return. The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.
* The MIN() function returns the smallest value of the selected column.
* The MAX() function returns the largest value of the selected column.

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

* The COUNT() function returns the number of rows that matches a specified criterion.

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

* The AVG() function returns the average value of a numeric column.

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

* The SUM() function returns the total sum of a numeric column.

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

* 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

WHERE CustomerName LIKE 'a%' Finds any values that start with "a"

WHERE CustomerName LIKE '%a' Finds any values that end with "a"

WHERE CustomerName LIKE '%or%' Finds any values that have "or" in any position

WHERE CustomerName LIKE '\_r%' Finds any values that have "r" in the second position

WHERE CustomerName LIKE 'a\_%' Finds any values that start with "a" and are at least 2 characters in length

WHERE CustomerName LIKE 'a\_\_%' Finds any values that start with "a" and are at least 3 characters in length

WHERE ContactName LIKE 'a%o' Finds any values that start with "a" and ends with "o"

SELECT \* FROM Customers  
WHERE CustomerName NOT LIKE 'a%';

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

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);

OR

SELECT \* FROM Customers  
WHERE Country NOT IN ('Germany', 'France', 'UK');

* 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.

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

SELECT \* FROM Products  
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'  
ORDER BY ProductName;

* 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.

SELECT column\_name AS alias\_name  
FROM table\_name;

Aliases can be useful when:

* There are more than one table involved in a query
* Functions are used in the query
* Column names are big or not very readable
* Two or more columns are combined together
* A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Joining 2 tables from Orders (table 1) customer id(column name) and customers(table 2) customer id

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers

ON Orders.CustomerID=Customers.CustomerID;

* INNER JOIN: Returns records that have matching values in both tables. The INNER JOIN keyword selects records that have matching values in both tables.

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

* LEFT JOIN: Returns all records from the left table, and the matched records from the right table

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID

* RIGHT JOIN: Returns all records from the right table, and the matched records from the left table. The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records (if any) from the left table (table1).

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

* CROSS JOIN: Returns all records from both tables. The CROSS JOIN keyword returns all matching records from both tables whether the other table matches or not. So, if there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

SELECT column\_name(s)  
FROM table1  
CROSS JOIN table2;

* A self-join is a regular join, but the table is joined with itself.

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City  
ORDER BY A.City;

* The UNION operator is used to combine the result-set of two or more SELECT statements.
* Every SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in every SELECT statement must also be in the same order

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

SELECT City, Country FROM Customers  
WHERE Country='Germany'  
UNION  
SELECT City, Country FROM Suppliers  
WHERE Country='Germany'  
ORDER BY City;

* The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country". 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.

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;

* The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

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

* The EXISTS operator is used to test for the existence of any record in a subquery. The EXISTS operator returns TRUE if the subquery returns one or more records.

SELECT column\_name(s)  
FROM table\_name  
WHERE EXISTS  
(SELECT column\_name FROM table\_name WHERE condition);

* The ANY and ALL operators allow you to perform a comparison between a single column value and a range of other values.

The ANY operator:

* returns a boolean value as a result
* returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for any of the values in the range.

SELECT ProductName  
FROM Products  
WHERE ProductID = ANY  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity = 10);

The ALL operator:

* returns a boolean value as a result
* returns TRUE if ALL of the subquery values meet the condition
* is used with SELECT, WHERE and HAVING statements

ALL means that the condition will be true only if the operation is true for all values in the range.

SELECT ProductName  
FROM Products  
WHERE ProductID = ALL  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity = 10);

* The INSERT INTO SELECT statement copies data from one table and inserts it into another table. The INSERT INTO SELECT statement requires that the data types in source and target tables matches.

INSERT INTO table2  
SELECT \* FROM table1WHERE condition;

INSERT INTO table2 (column1, column2, column3, ...)  
SELECT column1, column2, column3, ...  
FROM table1  
WHERE condition;

* The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause. If there is no ELSE part and no conditions are true, it returns NULL.

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

SELECT CustomerName, City, Country  
FROM Customers  
ORDER BY  
(CASE  
    WHEN City IS NULL THEN Country  
    ELSE City  
END);

* The MySQL [IFNULL()](https://www.w3schools.com/mysql/func_mysql_ifnull.asp) function lets you return an alternative value if an expression is NULL. Or we can use the [COALESCE()](https://www.w3schools.com/mysql/func_mysql_coalesce.asp) function

SELECT ProductName, UnitPrice \* (UnitsInStock + IFNULL(UnitsOnOrder, 0))  
FROM Products;

SELECT ProductName, UnitPrice \* (UnitsInStock + COALESCE(UnitsOnOrder, 0))  
FROM Products;

* Single line comments start with --. Multi-line comments start with /\* and end with \*/.
* Conditions

ALL TRUE if all of the subquery values meet the condition

AND TRUE if all the conditions separated by AND is TRUE

ANY TRUE if any of the subquery values meet the condition

BETWEEN TRUE if the operand is within the range of comparisons

EXISTS TRUE if the subquery returns one or more records

IN TRUE if the operand is equal to one of a list of expressions

LIKE TRUE if the operand matches a pattern

NOT Displays a record if the condition(s) is NOT TRUE

OR TRUE if any of the conditions separated by OR is TRUE

SOME TRUE if any of the subquery values meet the condition

* The CREATE DATABASE statement is used to create a new SQL database.

CREATE DATABASE databasename;

* The DROP DATABASE statement is used to drop an existing SQL database. Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!

DROP DATABASE databasename;

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

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

A copy of an existing table can also be created using CREATE TABLE.

CREATE TABLE new\_table\_name AS  
    SELECT column1, column2,...  
    FROM existing\_table\_name  
    WHERE ....;

* The following SQL statement drops the existing table "Shippers":

DROP TABLE Shippers;

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

TRUNCATE TABLE table\_name;

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

ALTER TABLE table\_name  
ADD column\_name datatype;

ALTER TABLE table\_name  
DROP COLUMN column\_name;

* Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement. SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table.

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

* [NOT NULL](https://www.w3schools.com/mysql/mysql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/mysql/mysql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/mysql/mysql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table. There can be only one primary key in a table.

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName)  
);

In the example above there is only ONE PRIMARY KEY (PK\_Person). However, the VALUE of the primary key is made up of TWO COLUMNS (ID + LastName).

* [FOREIGN KEY](https://www.w3schools.com/mysql/mysql_foreignkey.asp) - Prevents actions that would destroy links between tables

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

* [CHECK](https://www.w3schools.com/mysql/mysql_check.asp) - Ensures that the values in a column satisfies a specific condition

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CHECK (Age>=18)  
);

* [DEFAULT](https://www.w3schools.com/mysql/mysql_default.asp) - Sets a default value for a column if no value is specified

The DEFAULT constraint can also be used to insert system values, by using functions like [CURRENT\_DATE()](https://www.w3schools.com/mysql/func_mysql_current_date.asp):

CREATE TABLE Orders (  
    ID int NOT NULL,  
    OrderNumber int NOT NULL,  
    OrderDate date DEFAULT CURRENT\_DATE()  
);

* [CREATE INDEX](https://www.w3schools.com/mysql/mysql_create_index.asp) - Used to create and retrieve data from the database very quickly. Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

CREATE INDEX index\_name  
ON table\_name (column1, column2, ...);

* MySQL uses the AUTO\_INCREMENT keyword to perform an auto-increment feature. By default, the starting value for AUTO\_INCREMENT is 1, and it will increment by 1 for each new record.

CREATE TABLE Persons (  
    Personid int NOT NULL AUTO\_INCREMENT,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (Personid)  
);

MySQL comes with the following data types for storing a date or a date/time value in the database:

* DATE - format YYYY-MM-DD
* DATETIME - format: YYYY-MM-DD HH:MI:SS
* TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
* YEAR - format YYYY or YY
* In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. A view is created with the CREATE VIEW statement.

CREATE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

NOT IMPORTANT TOOL, USED AS SELECT ONLY

* The INSERT INTO statement is used to insert new records in a table. 2 ways to do this:

If changing the sequence of columns input

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

If you are adding values for all the columns of the table

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