**SQL: (STRUCTURED QUERY LANGUAGE)**

SQL is a standard language for storing, manipulating and retrieving data in database.

**What Can SQL Do:**

* SQL can create new database.
* SQL can create new tables in database.
* SQL can insert records in a database.
* SQL can delete records from a database.
* SQL can update records in a database.
* SQL can execute queries against a database.
* SQL can create stored procedures in a database.
* SQL can set permissions on tables, procedures and views.

**MYSQL: (MY STRUCTURED QUERY LANGUAGE)**

* MYSQL is a relational database system that uses SQL to query data from the database.
* MYSQL was developed by MYSQL AB but is currently acquired and owned by oracle corporation.
* MYSQL is software and not a programming language. Hence it does not have any commands or particular format.
* MYSQL is an open-source free platform that allows to any and every one.

**DBMS: (DATABASE MANAGEMENT SYSTEM)**

* It stores the data in the form of tables.
* Data base management system it is a software that is used to define, create maintain a database and provides controlled access to the data.
* It stores data the files.
* Data store generally is either hierarchical form or a navigational form.
* No relationship between data. It supports the single user.
* It is used for small organization and small data.

**RDBMS: (RELATIONAL DATABASE MANAGEMENT SYSTEM)**

* It is an advance version of the DBMS.
* It is store data a tabular forms.it is used in tabular structure.
* Data is in a tabular form it is relationship between each other.
* It supports the multiple users. It handles large amount of data.

How to create Data Base:

create database database\_name;

ex: create database Kalyani;

How to use database:

Use database\_name;

ex: use Kalyani;

How show the database or tables:-

show databases and show table;

**Data Types:**

**Int:** It is used to store integer type data. Like Em id, age, etc…

**Big int:** It is used to store to big int like adhara number, phone number…

**Money:** It is used to store decimal value like saraly, petrol cost…

**Char:** It is used to store character data like gender, grade….

**Varchar:** It is used to store character data it is variable length like ename, typename …

**Date**: It is used to store datatypes values like DOB…..

* 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

How to create table :

CREATE TABLE table\_name (

column1\_name column1\_datatype constraints,

column2\_name column2\_datatype constraints,);

ex:- CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

BirthDate DATE,

Position VARCHAR(50));

INSERT:

INSERT INTO Employees (EmployeeID, FirstName, LastName, BirthDate, HireDate, Position, Salary)

VALUES (1, 'John', 'Doe', '1980-05-15', '2020-06-01', 'Manager', 75000.00);

**DELETE:**

It is used to remove one or more rows from a table based on specified conditions. **Syntax**: DELETE FROM table\_name WHERE condition;

ex: DELETE FROM Employees WHERE Employee ID = 1;

**Drop:**

It is used to specified object is permanently removed,

**Syntax:** DROP TABLE table\_name;

Ex: DROP TABLE Employees;

**TRUNCATE:**

The TRUNCATE statement in SQL is used to remove all rows from a table.

**Syntax**: TRUNCATE TABLE table\_name;

ex: TRUNCATE TABLE Employees;

**UPDATE:**

It is used to modify existing data in a table. It allows you to update one or more rows in a table based on specified conditions.

**Syntax:**

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

EX: UPDATE Employees SET Salary = 80000 WHERE EmployeeID = 1;

**ALTER:**

It is modify the structure of an existing database object, such as a table. You can use it to add, delete, or modify columns in a table.

**Syntax:** ALTER TABLE table\_name ADD column\_name datatype;

Ex: ALTER TABLE Employees **ADD** Email VARCHAR(100);

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

**Syntax:** ALTER TABLE table\_name **MODIFY** COLUMN column\_name new\_datatype;

**PRIMARY KEY:**

* The primary key constraint uniquely identifies each record in a table.
* primary key must contain “unique values” and cannot contain null values.
* A table can have only one primary key.

Ex: create table student (sid int primary key, sname varchar(10));

**FOREGIN KEY:**

A foreign key is a field or a column that is used to establish a link between 2 tables.

A foreign key in one table used to point primary key in another table.

EX:

Create table statesinfo (sid int primary key identity,sname VARCHAR(10),SCODE VARCHAR(20));

INSERT into statesinfo values(‘andhrapradesh’,ap’);

INSERT into statesinfo values(‘guntur’,gnt’);

select\*from statesinfo;

Create table districtinfo (did int primary key,dname varchar(50),sid int references stateinfo(sid));

**NOTE:** The foreign key constraint prevents invalid data from being inserted into the foreign key column because it has to be one of the values contained in the parent table.

(or)

CREATE TABLE child\_table (

column1 datatype,

column2 datatype,

...

FOREIGN KEY (column\_name) REFERENCES parent\_table (parent\_column)

);

**IDENTITY:-**

Identity allows a unique number to be generated automatically when a new record is inserted into table.

The starting value for identity is 1 an it will increment by 1.for each new record.

**Note:-**

To specify that the “emid” column should start at value 100 an increment by 5 change it to identity(100,5).

Ex:-create table table student (eid int primary key identity (100,5)ename varchar(10));

Insert into emp values(‘kalyani);

**AUTO-INCREMENT:-**

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

EX:-

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

**SELECT STATEMENT :-**

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

**SYNTAX**:-SELECT column1, column2, ...FROM table\_name WHERE condition;

FOR ALL COLUMNS:- SELECT \* FROM Employees;

**SELECT DISTINCT:-**

The select distinct statement is used to return only distinct(different) values. It is used to retrieve unique values from a column or a combination of columns, eliminating any duplicate entries.

**SYNTAX:-**Select distinct col1,col2….. from tablename;

Ex:- SELECT DISTINCT Position FROM Employees;

**WHERE CLAUSE**:-

There where clause is used filter records. It is used to extract only records that fulfill a specified condition.

**SYNTAX**:-SELECT column1, column2, ...FROM table\_name WHERE condition;

EX:- SELECT FirstName, LastName, Salary FROM Employees WHERE Salary > 50000;

**NOTE**:-The where clause is not only used in select statements ,it is also used in update ,delete .

**ORDER BY:-**

The order by keyword is used to sort the result set in ascending or descending order.

The order by keyword sorts the records in ascending order by default.To sort the records in descending order, use the desc keyword.

**SYNTAX**:- SELECT column1, column2, ...FROM table\_nameORDER BY column\_name [ASC|DESC];

ex:- SELECT FirstName, LastName, Salary FROM Employees ORDER BY LastName ASC;

**AND ,OR,NOT OPERTORS :-**

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.

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 ...;

EX:- SELECT FirstName, LastName, Department, Salary FROM Employees

WHERE Department = 'Sales' AND Salary > 50000;

**OR:-**

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 ;

EX:-SELECT FirstName, LastName, Department, Salary FROM Employees

WHERE Department = 'Sales' OR Salary > 50000;

**NOT:-**

The not operator displays a record if the conditions is not true.

**SYNTAX**:-SELECT column1, column2, ...FROM table\_name WHERE NOT condition;

EX:- SELECT FirstName, LastName, Department FROM Employees

WHERE NOT Department = 'Sales';

**AGGREATE FUNCTIONS:-**

**Min():-**To find the smallest value in the selected column

Ex:-select min (salary) from employee.

**Max:-**To find the highest value in the selected column

Ex:-select max(salary) from employee.

**Count():-**To find total no.of rows based on the condition.

Ex:-select count(emid) from employee where country=’uk’;

**Avg():-**To calculate average value in the numeric column.

Ex:-select avg(salary) from employee.

**Sum():-**To calculate total sum of a numeric column

Ex:-select sum(salary) from employee where country=’uk’;

**LIKE OPERATOR AND WILD CARDS**:-

The like operator is used in a where clause to search for a specified pattern in a column.

There are 2 wildcards often used in conjunction with the like operator(%&\_).

A wild card character is used to substitute one or more character in a string.

Wildcards character are used with the like operator.

* % represents zero or more characters.\_
* \_represents a single character.
* [] represents any single character with in the brackets.
* \*represents any character not in the brackets.

**SYNTAX:-** SELECT column1, column2, ...FROM table\_name WHERE column\_name LIKE pattern;

EX:-Select \*from employee where email like’%us.com’.

EX:- Select \*from employee where email like’-p’.

EX:- Select \*from employee where email like’[jm]’.

EX:- Select \*from employee where email like’[^jm]%’

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.

EX:-select\*from employee where age=36 or age=32 or age=22;

**SYNTAX:-**Select\*from employee where age in(36,32,34)

BETWEEN OPERATOR:-

The between operator selects values within a given range.

EX:- Select\*from employee where age between 25 and 30;

**NOTE**:- The between operator is inclusive:begin and end values are included.

**GROUP BY:-**

The group by clause is used to group rows by one or more columns.

Note:The group by clause is used in conjunction with aggregate function such as min(),max(),sum(),avg() and count()etc..

**SYNTAX**:- SELECT column1, aggregate\_function(column2) FROM table\_name

WHERE condition GROUP BY column1;

Ex:-select country sum (salary)as total salary from employee group by county.

**HAVING:-**

The having clause was added to sql because the where keyword cannot be used with aggregate function. we can use the group by clause with having clause to filter the result set based on aggregate functions.

**SYNTAX**:-SELECT column1, aggregate\_function(column2) FROM table\_name GROUP BY column1

HAVING aggregate\_function(column2) condition;

Ex:- SELECT Department, COUNT(\*) AS EmployeeCount FROM Employees GROUP BY Department

HAVING COUNT(\*) > 5;

JOINS:-

It is used to retrieve data from 2 or more related tables. In genral tables are related to each using foreign key.

EX:SELECT \*from userinfo join statesinfo on userinfo.stateid=statesinfo.stateid.

Here are the different types of the JOINs :

* (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:-

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