MySQL®

Let's start learning..

Table of Contents

Note: Click on the page number given in front to go to any topic.

What is SQL?

- SQL (Structured Query Language) is a standard language for storing, manipulating and retrieving data in databases.
- SQL is not a database system, It is language which is used by database management system like (Oracle, MySQL, Microsoft SQL Server, MongoDB and more).
- It is also pronounced as See-Quell.
- SQL language is mainly designed for maintaining the data in relational database management systems (RDBMS).
- Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.
- Programming languages like php, asp.net and python use sql for backend.

What is data?

- Data is a collection of information.
- Data can be name, age, height, phone number, picture, PDF, audio and more.

What is database?

- A database is an electronic storage which is created by DBMS software's that stores the organized collection of records. It can be accessed and manage by the user very easily. It allows us to organize data into tables, rows, columns, and indexes to find the relevant information very quickly.

What is table?

- Table is a collection of data which is organized in terms of rows and columns.

Employee_ID	Employee_Name	City	Mobile Number
1	AJAY	Bhopal	1111-2222-33
2	SUNIL	Jaipur	3311-2222-33
3	MANOJ	Udaipur	2211-2222-33
4	KOMAL	SURAT	2211-2222-33

What is field?

- A field is a data structure for a single piece of data. For example, in a table called student contact information, telephone number is a field in column.

Employee_ID	Employee_Name	City	Mobile Number
1	AJAY	Bhopal	1111-2222-33
2	SUNIL	Jaipur	3311-2222-33
3	MANOJ	Udaipur	2211-2222-33
4	KOMAL	SURAT	2211-2222-33

What is record?

- In relational databases, a record is a group of related data held within the same structure.

Employee_ID	Employee_Name	City	Mobile Number
1	AJAY	Bhopal	1111-2222-33
2	SUNIL	Jaipur	3311-2222-33
3	MANOJ	Udaipur	2211-2222-33
4	KOMAL	SURAT	2211-2222-33

How to work with SQL?

To working with SQL you can use any database management system but in this tutorial I am using XAMPP and MySQL Workbench. So first you have to download and install XAMPP and MySQL Workbench.

Download Link For XAMPP -
https://www.apachefriends.org/download.html

Download Link For MySQL Workbench-
https://www.mysql.com/products/workbench/

To connect to MySQL from the command line, follow these:

Syntax: c:\xampp\mysql\bin>mysql -u user -p password -h localhost IP

Example: c:\xampp\mysql\bin>mysql -u root -p 12345 -h 127.0.0.1

Create Database

- **Syntax :** CREATE DATABASE database_name;
- **Example :** CREATE DATABASE company;

Select Database

- **Syntax :** USE database_name;
- **Example :** USE company;

Show Database

- **Syntax :** SHOW DATABASES;

Drop Database

- **Syntax :** DROP DATABASE database_name;

Create Table Statement

- **Syntax:**

```
 CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Example:

```
CREATE TABLE employeeInfo(
ID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
Emp_name varchar(30) NOT NULL,
Address varchar(255) NOT NULL,
City varchar(50) NOT NULL,
Age int NOT NULL,
```

DOJ date **NOT NULL**,
Designation varchar(50) **NOT NULL**,
Salary decimal(15,2) **NOT NULL**,
Mobile varchar(10) **NOT NULL**);

Drop/Delete Table Statement

The **DROP TABLE** statement is used to remove a table for database.

Syntax: DROP TABLE table_name;

ALTER TABLE Statement

- This statement is used to add, delete, or modify columns in an existing table.
- This is also used to add and drop various constraints on an existing table.

Add Column

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example:-

```
ALTER TABLE employeeInfo
ADD Email varchar(255);
```

DROP Column

```
ALTER TABLE table_name;
DROP column_name;
```

Example:-

```
ALTER TABLE employeeInfo;
DROP Email;
```

MYSQL INSERT INTO Statement

The **INSERT INTO** Statement is used to add new data in the table.

Syntax:

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

Example: (Inserting record in table)

```
INSERT INTO employeeInfo (Emp_name,  Address,  City,  Age,  DOJ,
Designation,  Salary,  Mobile)
VALUES ('Sunil Kumar', '104, Street No. 13', 'Jaipur', 29,'2020-5-3',
'Manager',  42000, '9188822200');

INSERT INTO employeeInfo (Emp_name,  Address, City,  Age,  DOJ,
Designation,  Salary,  Mobile)
VALUES ('Manoj Singh', '72, Street No. 1', 'Jaipur', 30, '2021-2-1',
'Programmer', 43000, '9188822211');

INSERT INTO employeeInfo (Emp_name,  Address,  City,  Age,  DOJ,
Designation,  Salary,  Mobile)
VALUES ('Anil Kumar', 'House No.53, Street No.7', 'Udaipur', 32,'2021-3-
1', 'Programmer',  42000, '9133322211');
```

INSERT INTO employeeInfo (Emp_name, Address, City, Age, DOJ, Designation, Salary, Mobile)
VALUES ('Kamal', 'P76, Block No.5', 'Jaipur', 36, '2020-2-1', 'Sr. Programmer', 60000, '9188833311');

INSERT INTO employeeInfo (Emp_name, Address, City, Age, DOJ, Designation, Salary, Mobile)
VALUES ('Daanish', 'RK Nagar, New Bus Stand', 'Ajmer', 37, '2021-4-1', 'Sr. Programmer', 60000, '9188822255');

SQL SELECT Statement

The **SELECT** statement is used to selecting (showing) the data from a table.

<u>To selecting whole data from table</u>

Syntax: SELECT *FROM table_name;
Example: SELECT *FROM employeeInfo;

Syntax: SELECT column1, column2, columnN FROM table_name;
Example: SELECT Emp_name, Mobile from employeeInfo;

The SQL WHERE Clause

The **WHERE** clause is used to filter records from table.

Syntax:
SELECT column1, column2, columnN
FROM table_name

```
WHERE Designation = [Condition];
```

Example: 1

```
SELECT Emp_name, Mobile
FROM employeeInfo
WHERE Designation = 'Programmer';
```

Example: 2

```
SELECT Emp_name, Mobile, Designation, Salary
FROM employeeInfo
WHERE Salary>45000;
```

The SQL UPDATE

If you want to modify the existing records in a table then you can use the query.

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2....
WHERE [condition];
```

<u>Example</u>: 1

```
UPDATE employeeInfo
SET Address =
WHERE [condition];
```

<u>Example</u>: 2

```
UPDATE employeeInfo
SET Address = 'P79, Block No.5'
WHERE ID=4;
```

Example: 3

```
UPDATE employeeInfo
SET Salary = 45000
WHERE ID = 5;
```

The SQL DELETE

The SQL DELETE Query is used to delete the existing records from a table.

Syntax:

```
DELETE FROM table_name
WHERE [condition];
```

Example:

```
DELETE FROM employeeInfo
WHERE ID = 5;
```

If you want to delete all records from table then you can use

```
DELETE FROM TABLE_NAME;
```

The SQL AND OPERATOR

You can use this operator for checking multiple conditions.

Syntax:

```
SELECT column1, column2,....
FROM table_name
WHERE [condition1] AND [condition2]...AND [conditionN];
```

Example: 1

```
SELECT ID, Emp_name
FROM employeeInfo
WHERE Designation = 'Programmer' AND Age > 30;
```

Example: 2

```
SELECT ID, Emp_name
FROM employeeInfo
WHERE Designation = 'Programmer' AND City = 'Jaipur';
```

The SQL OR OPERATOR

Syntax:

```
SELECT column1, column2,....
FROM table_name
WHERE [condition1] OR [condition2]...OR [conditionN];
```

Example:

```
SELECT ID, Emp_name, City
FROM employeeInfo
WHERE City = 'Jaipur' OR Salary = 42000;
```

The SQL NOT OPERATOR

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Example:

SELECT ID, Emp_name, City

FROM employeeInfo

WHERE NOT City = 'Jaipur';

The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

- 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;

Example: 1

SELECT *FROM employeeInfo

WHERE City LIKE 'J%'; **(Start With) you can also use (End With Like – %J)**

Example: 2

SELECT *FROM employeeInfo
WHERE Mobile LIKE '__8%';

Example: 3

SELECT * FROM employeeInfo

WHERE Emp_name NOT LIKE 'm%';

The SQL IN Operator

Example: 1

SELECT * FROM employeeInfo

WHERE City IN ('Udaypur', 'Ajmer');

Example: 2

SELECT * FROM employeeInfo

WHERE City NOT IN ('Udaypur', 'Ajmer');

The SQL BETWEEN Operator

The BETWEEN operator selects values within a given range.

Example: 1

SELECT * FROM employeeInfo

WHERE Salary **BETWEEN** 30000 AND 50000;

Example: 2

SELECT * FROM employeeInfo

WHERE ID **BETWEEN** 3 AND 5;

Example: 3

SELECT * FROM employeeInfo

WHERE DOJ **BETWEEN** '2021-01-01' AND '2021-05-30';

The SQL SELECT TOP Clause

SELECT TOP Clause is used to show specify the number of records from table.

Note: Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause.

Syntax:

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE [condition];
```

Example: (Note: Only For MySQL)

```
SELECT * FROM employeeInfo
LIMIT 3;
```

The SQL ORDER BY Clause

The SQL **ORDER BY** clause is used to sort the column data in ascending or descending order.

Example: 1

```
SELECT * FROM employeeInfo
ORDER BY Emp_name, Salary;
```

Example: 2

```
SELECT * FROM employeeInfo
ORDER BY Emp_name DESC;
```

Working with SQL functions

COUNT() Example:

SELECT COUNT(Emp_name)

FROM employeeInfo;

SUM() Example:

SELECT SUM(Salary)

FROM employeeInfo;

AVG() Example:

SELECT AVG(Salary)

FROM employeeInfo;

MIN() Example:

SELECT Emp_name, MIN(Salary) AS LowestSalary

FROM employeeInfo;

MAX() Example:

SELECT Emp_name, MIN(Salary) AS HighestSalary

FROM employeeInfo;

SQL Joins

The SQL **Joins** clause is used to combine records from two or more tables.

Example:

Step 1 (Create database)

```
CREATE DATABASE companydb;
```

Step 2 (Select database)

```
USE  companydb;
```

Step 3 (Create table)

```
CREATE TABLE Customer
(
 CustomerID int primary key,
 CustomerName varchar(20),
 City varchar(20)
);
```

Step 4 (Insert records in table)

```
INSERT INTO Customer VALUES(101,'Ashish','Kota');
INSERT INTO Customer VALUES(102,'Ajay','Ajmer');
INSERT INTO Customer VALUES(103,'Jay','Bundi');
INSERT INTO Customer VALUES(104,'Aman','Jhalawar');
INSERT INTO Customer VALUES(105,'Chirag','Udaipur');
INSERT INTO Customer VALUES(106,'Deepak','Jodhpur');
INSERT INTO Customer VALUES(107,'Rohan','Jaipur');
INSERT INTO Customer VALUES(108,'Dinesh','Alwar');
INSERT INTO Customer VALUES(109,'Suresh','Kota');
INSERT INTO Customer VALUES(110,'Ankit','Jaipur');
```

Step 5 (Create IInd table)

```
CREATE TABLE Orders
(
OrderID int,
CustomerID int primary key,
OrderDate date
);
```

Step 6 (Insert records in this table)

```
INSERT INTO Orders VALUES (1,101,'2021-01-05');
INSERT INTO Orders VALUES (2,102,'2021-02-10');
INSERT INTO Orders VALUES (3,103,'2021-01-04');
INSERT INTO Orders VALUES (4,104,'2021-03-08');
INSERT INTO Orders VALUES (5,105,'2021-04-09');
INSERT INTO Orders VALUES (6,106,'2021-01-05');
INSERT INTO Orders VALUES (7,107,'2021-05-12');
INSERT INTO Orders VALUES (8,108,'2021-02-05');
INSERT INTO Orders VALUES (9,109,'2021-06-03');
INSERT INTO Orders VALUES (10,110,'2021-07-04');
```

Step 7 (Check both tables)

```
SELECT *FROM Customer;

SELECT * FROM Orders;
```

<u>INNER JOIN QUERY</u> 1

```
SELECT Orders.OrderID, Customer.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customer ON Orders.CustomerID=Customer.CustomerID;
```

<u>INNER JOIN QUERY</u> 2

```
SELECT Orders.OrderDate, Customer.City
FROM Orders
INNER JOIN Customer ON Orders.CustomerID=Customer.CustomerID;
```

Some other SQL Joins Like – LEFT JOIN, RIGHT JOIN, FULL JOIN and SELF JOIN, you can read from Google.

The SQL GROUP BY

The GROUP BY statement groups rows that have the same values into rows. We can also use these functions
(COUNT(), MAX(), MIN(), SUM(), AVG()) with group by statements.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s);
```

Example:

```
SELECT COUNT(Designation), SUM(Salary), Designation
From employeeInfo
GROUP BY Designation;
```

SQL INSERT INTO SELECT Statement

You can copies data from one table and inserts it into another table.

Syntax:

```
INSERT INTO New_Table
SELECT * FROM Old_Table
WHERE condition;
```

Example:

```
INSERT INTO employeeInfoBackup
SELECT * FROM employeeInfo;
```

The SQL TRUNCATE TABLE Command

If you want to delete complete data from an existing table then you can use this command. If you use DROP Table command it would remove complete table structure form the database.

Syntax:

```
TRUNCATE TABLE  table_name;
```

SQL Constraints

By using SQL constraints you can set specify rules for the data in a table.

The following constraints are commonly used in SQL:

NOT NULL CONSTRAINT- Ensures that a column value should not NULL.

Example of NOT NULL Constraints:

Step 1 Create new database

```
CREATE DATABASE constraintsExample;
```

Step 2 Select this database

```
USE constraintsExample;
```

Step 3 Create Table

```
CREATE TABLE notnullExample
(
EmpName varchar(30),
Age int
);
```

Step 4 Insert Records

```
INSERT INTO notnullExample (EmpName, Age) VALUES ('Raju', 26);
INSERT INTO notnullExample (Age) VALUES (26);
```

Step 5 View Table

```
Select * from notnullExample;
```

Step 6 Delete Table

```
DROP TABLE notnullExample;
```

Step 7 Create Table Again

```
CREATE TABLE notnullExample
(
EmpName varchar(30) NOT NULL,
Age int
);
```

Step 8 Insert Records

```
INSERT INTO notnullExample (EmpName, Age)  VALUES ('Raju', 26);
INSERT INTO notnullExample (Age)  VALUES (26);
```

Step 9 View Table

```
Select * from notnullExample;
```

DEFAULT CONSTRAINT - Set a default value for a column if no value is inserting.

Example of DEFAULT Constraints:

Step 1 Create Table

```
CREATE TABLE defaultExample
(
EmpName varchar(30) NOT NULL,
Age int,
Mobile varchar(50) default 'Mobile no. is not available'
);
```

Step 2 Insert Data

```
INSERT INTO defaultExample (EmpName, Age, Mobile)
VALUES ('Raju', 26, '001223665');
```

Step 3 Insert this Record

Note: - To checking DEFAULT constraint insert this record in the table then 'Mobile no. is not available' this default value will store in mobile number column because of DEFAULT Constraint.

```
INSERT INTO defaultExample (EmpName, Age)
VALUES ('Raju', 26);
```

Step 4 View Table

```
SELECT * FROM defaultExample;
```

CHECK CONSTRAINT - You can define a CHECK constraint on a column it will allow only certain values for this column. If condition is matched then value will store in column.

Step 1 Creat Table

```
CREATE TABLE checkExample
(
EmpName varchar(255),
Age int,
CHECK (Age>=18)
);
```

Step 2 Insert Record

```
INSERT INTO checkExample (EmpName, Age)  VALUES ('Raju', 26);
```

Step 3 Insert this Record

Note: - To checking CHECK constraint insert this record in the table then you will find error because in this record you are inserting age 17 which is less than 18.

```
INSERT INTO checkExample (EmpName, Age)  VALUES ('Raju', 17);
```

UNIQUE CONSTRAINT - The UNIQUE constraint ensures that all values in a column are different.

Example of UNIQUE CONSTRAINT

Step 1 Create Table

```
CREATE TABLE uniqueExample (
EmpName varchar(255) NOT NULL,
Mobile varchar(10) NOT NULL UNIQUE
);
```

Step 2 Insert Record

```
INSERT INTO uniqueExample (EmpName, Mobile)  VALUES ('Raju',
'0123456789');
```

Step 3 Insert this Record

Note: - To checking UNIQUE Constraint insert this record in table then you will find error because in this record mobile number should be different and you are again inserting same mobile number.

```
INSERT INTO uniqueExample (EmpName, Mobile)  VALUES ('Sunil',
'0123456789');
```

PRIMARY KEY CONSTRAINT - Primary keys column must contain UNIQUE values, and cannot contain NULL values.

SQL FOREIGN KEY CONSTRAINT -

Step 1 Create table

```
CREATE TABLE Department
(
Dept_Id int primary key,Dept_Name varchar(50)
);
```

Step 2 Insert Records

```
INSERT INTO Department VALUES (1,'Account');
INSERT INTO Department VALUES (2,'HR');
INSERT INTO Department VALUES (3,'IT');
```

Step 3 Create IInd table

```
CREATE TABLE Employee_Details
(
Emp_ID int primary key,
Emp_Name varchar(50) Not null,
Dept_Id int,
FOREIGN KEY (Dept_Id) REFERENCES Department(Dept_Id)
);
```

Step 4 Insert Records

```
INSERT INTO Employee_Details VALUES (1,'Raj',1);
INSERT INTO Employee_Details VALUES (2,'Rahul',2);
INSERT INTO Employee_Details VALUES (3,'Kunal',3);
```

```
SELECT * FROM Employee_Details;
```

Note: - To checking FOREIGN KEY Constraint insert this record in table then you will find error because Dept_Id will not match.

Step 5 Insert this Records

```
INSERT INTO Employee_Details VALUES (4, 'sohan', 4);
```

The End