**SQL** is a database computer language designed for the retrieval and management of data in a relational database. **SQL** stands for **Structured Query Language**. This tutorial will give you a quick start to SQL. It covers most of the topics required for a basic understanding of SQL and to get a feel of how it works.

Why to Learn SQL?

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as −

* MS SQL Server using T-SQL,
* Oracle using PL/SQL,
* MS Access version of SQL is called JET SQL (native format) etc.

Applications of SQL

As mentioned before, SQL is one of the most widely used query language over the databases. I'm going to list few of them here:

* Allows users to access data in the relational database management systems.
* Allows users to describe the data.
* Allows users to define the data in a database and manipulate that data.
* Allows to embed within other languages using SQL modules, libraries & pre-compilers.
* Allows users to create and drop databases and tables.
* Allows users to create view, stored procedure, functions in a database.
* Allows users to set permissions on tables, procedures and views.

## Audience

This **SQL tutorial** is prepared for beginners to help them understand the basic as well as the advanced concepts related to SQL languages. This tutorial will give you enough understanding on the various components of SQL along with suitable examples.

## Prerequisites

Before you start practicing with various types of examples given in this tutorial, I am assuming that you are already aware about what a database is, especially the RDBMS and what is a computer programming language.

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

## Syntax

The basic syntax of this CREATE DATABASE statement is as follows −

CREATE DATABASE DatabaseName;

Always the database name should be unique within the RDBMS.

## Example

If you want to create a new database <testDB>, then the CREATE DATABASE statement would be as shown below −

SQL> CREATE DATABASE testDB;

Make sure you have the admin privilege before creating any database. Once a database is created, you can check it in the list of databases as follows −

SQL> SHOW DATABASES;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| AMROOD |

| TUTORIALSPOINT |

| mysql |

| orig |

| test |

| testDB |

+--------------------+

7 rows in set (0.00 sec)

The SQL **DROP DATABASE** statement is used to drop an existing database in SQL schema.

## Syntax

The basic syntax of DROP DATABASE statement is as follows −

DROP DATABASE DatabaseName;

Always the database name should be unique within the RDBMS.

## Example

If you want to delete an existing database <testDB>, then the DROP DATABASE statement would be as shown below −

SQL> DROP DATABASE testDB;

**NOTE** − Be careful before using this operation because by deleting an existing database would result in loss of complete information stored in the database.

Make sure you have the admin privilege before dropping any database. Once a database is dropped, you can check it in the list of the databases as shown below −

SQL> SHOW DATABASES;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| AMROOD |

| TUTORIALSPOINT |

| mysql |

| orig |

| test |

+--------------------+

6 rows in set (0.00 sec)

The SQL **USE** statement is used to select any existing database in the SQL schema.

## Syntax

The basic syntax of the USE statement is as shown below −

USE DatabaseName;

Always the database name should be unique within the RDBMS.

## Example

You can check the available databases as shown below −

SQL> SHOW DATABASES;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| AMROOD |

| TUTORIALSPOINT |

| mysql |

| orig |

| test |

+--------------------+

6 rows in set (0.00 sec)

Now, if you want to work with the AMROOD database, then you can execute the following SQL command and start working with the AMROOD database.

SQL> USE AMROOD;

Creating a basic table involves naming the table and defining its columns and each column's data type.

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

## Syntax

|  |  |  |
| --- | --- | --- |
| Eno | EName | ESalary |
| 101 | Abi | 10000 |
| 102 | Akash | 15000 |

The basic syntax of the CREATE TABLE statement is as follows −

CREATE TABLE table\_name(

column1 datatype,

column2 datatype,

column3 datatype,

.....

columnN datatype,

PRIMARY KEY( one or more columns )

);

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is. The syntax becomes clearer with the following example.

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement. You can check the complete details at [Create Table Using another Table.](https://www.tutorialspoint.com/sql/sql-create-table-using-tables.htm)

## Example

The following code block is an example, which creates a CUSTOMERS table with an ID as a primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table −

SQL> CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

You can verify if your table has been created successfully by looking at the message displayed by the SQL server, otherwise you can use the **DESC** command as follows −

SQL> DESC CUSTOMERS;

+---------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+---------------+------+-----+---------+-------+

| ID | int(11) | NO | PRI | | |

| NAME | varchar(20) | NO | | | |

| AGE | int(11) | NO | | | |

| ADDRESS | char(25) | YES | | NULL | |

| SALARY | decimal(18,2) | YES | | NULL | |

+---------+---------------+------+-----+---------+-------+

5 rows in set (0.00 sec)

Now, you have CUSTOMERS table available in your database which you can use to store the required information related to customers.

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

**NOTE** − You should be very careful while using this command because once a table is deleted then all the information available in that table will also be lost forever.

## Syntax

The basic syntax of this DROP TABLE statement is as follows −

DROP TABLE table\_name;

## Example

Let us first verify the CUSTOMERS table and then we will delete it from the database as shown below −

SQL> DESC CUSTOMERS;

+---------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+---------------+------+-----+---------+-------+

| ID | int(11) | NO | PRI | | |

| NAME | varchar(20) | NO | | | |

| AGE | int(11) | NO | | | |

| ADDRESS | char(25) | YES | | NULL | |

| SALARY | decimal(18,2) | YES | | NULL | |

+---------+---------------+------+-----+---------+-------+

5 rows in set (0.00 sec)

This means that the CUSTOMERS table is available in the database, so let us now drop it as shown below.

SQL> DROP TABLE CUSTOMERS;

Query OK, 0 rows affected (0.01 sec)

Now, if you would try the DESC command, then you will get the following error −

SQL> DESC CUSTOMERS;

ERROR 1146 (42S02): Table 'TEST.CUSTOMERS' doesn't exist

Here, TEST is the database name which we are using for our examples.

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

### **Syntax**

There are two basic syntaxes of the INSERT INTO statement which are shown below.

INSERT INTO TABLE\_NAME (column1, column2, column3,...columnN)

VALUES (value1, value2, value3,...valueN);

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The **SQL INSERT INTO** syntax will be as follows −

INSERT INTO TABLE\_NAME VALUES (value1,value2,value3,...valueN);

### **Example**

The following statements would create six records in the CUSTOMERS table.

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (6, 'Komal', 22, 'MP', 4500.00 );

You can create a record in the CUSTOMERS table by using the second syntax as shown below.

INSERT INTO CUSTOMERS

VALUES (7, 'Muffy', 24, 'Indore', 10000.00 );

All the above statements would produce the following records in the CUSTOMERS table as shown below.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

## Populate one table using another table

You can populate the data into a table through the select statement over another table; provided the other table has a set of fields, which are required to populate the first table.

Here is the syntax −

INSERT INTO first\_table\_name [(column1, column2, ... columnN)]

SELECT column1, column2, ...columnN

FROM second\_table\_name

[WHERE condition];

INSERT INTO CUSTOMERS [(ID, NAME)]

SELECT ID, NAME FROM PURCHASE

WHERE (ID =101)

INSERT INTO Customers (CustomerName, City, Country)  
SELECT SupplierName, City, Country FROM Suppliers  
WHERE Country='Germany';

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **CustomerName** | **ContactName** | **Address** | **City** | **PostalCode** | **Country** |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |

And a selection from the "Suppliers" table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SupplierID** | **SupplierName** | **ContactName** | **Address** | **City** | **Postal Code** | **Country** |
| 1 | Exotic Liquid | Charlotte Cooper | 49 Gilbert St. | Londona | EC1 4SD | UK |
| 2 | New Orleans Cajun Delights | Shelley Burke | P.O. Box 78934 | New Orleans | 70117 | USA |
| 3 | Grandma Kelly's Homestead | Regina Murphy | 707 Oxford Rd. | Ann Arbor | 481 |  |

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

## Syntax

The basic syntax of the SELECT statement is as follows −

SELECT column1, column2, columnN FROM table\_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

SELECT \* FROM table\_name;

## Example

SELECT \* FROM CUSTOMER;

Consider the CUSTOMERS table having the following records −

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

The following code is an example, which would fetch the ID, Name and Salary fields of the customers available in CUSTOMERS table.

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;

This would produce the following result −

+----+----------+----------+

| ID | NAME | SALARY |

+----+----------+----------+

| 1 | Ramesh | 2000.00 |

| 2 | Khilan | 1500.00 |

| 3 | kaushik | 2000.00 |

| 4 | Chaitali | 6500.00 |

| 5 | Hardik | 8500.00 |

| 6 | Komal | 4500.00 |

| 7 | Muffy | 10000.00 |

+----+----------+----------+

If you want to fetch all the fields of the CUSTOMERS table, then you should use the following query.

SQL> SELECT \* FROM CUSTOMERS;

This would produce the result as shown below.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | Delhi | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

SELECT NAME FROM CUSTOMERS WHERE ADDRESS = Delhi

Chaitali

Hardik

Komal

Muffy

The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetching only the necessary records.

The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc., which we would examine in the subsequent chapters.

## Syntax

The basic syntax of the SELECT statement with the WHERE clause is as shown below.

SELECT column1, column2, columnN

FROM table\_name

WHERE [condition]

You can specify a condition using the [comparison or logical operators](https://www.tutorialspoint.com/sql/sql-operators.htm) like >, <, =, **LIKE, NOT**, etc. The following examples would make this concept clear.

## Example

Consider the CUSTOMERS table having the following records −

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Mumbai | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | MP | 10000.00 |

+----+----------+-----+-----------+----------+

The following code is an example which would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 −

SQL> SELECT ID, NAME, SALARY

FROM CUSTOMERS

WHERE SALARY > 2000;

This would produce the following result −

+----+----------+----------+

| ID | NAME | SALARY |

+----+----------+----------+

| 4 | Chaitali | 6500.00 |

| 5 | Hardik | 8500.00 |

| 6 | Komal | 4500.00 |

| 7 | Muffy | 10000.00 |

+----+----------+----------+

The following query is an example, which would fetch the ID, Name and Salary fields from the CUSTOMERS table for a customer with the name **Hardik**.

Here, it is important to note that all the strings should be given inside single quotes (''). Whereas, numeric values should be given without any quote as in the above example.

SQL> SELECT ID, NAME, SALARY

FROM CUSTOMERS

WHERE NAME = 'Hardik';

This would produce the following result −

+----+----------+----------+

| ID | NAME | SALARY |

+----+----------+----------+

| 5 | Hardik | 8500.00 |

+----+----------+----------+

**The SQL SELECT DISTINCT Statement**

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

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

SELECT DISTINCT column1, column2, ...

FROM table\_name;

SELECT DISTINCT ADDRESS FROM CUSTOMERS;

SELECT COUNT(DISTINCT Country) FROM Customers;

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | COUNTRY | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Mumbai | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | MP | 10000.00 |

+----+----------+-----+-----------+----------+

3

Operators in The WHERE Clause

The following operators can be used in the WHERE clause:

Operator Description

= Equal

> Greater than

< Less than

>= Greater than or equal

<= Less than or equal

<> Not equal. Note: In some versions of SQL this operator may be written as !=

BETWEEN Between a certain range

LIKE Search for a pattern

IN To specify multiple possible values for a column

EXAMPLE:

SELECT \* FROM Products

WHERE Price BETWEEN 50 AND 60;

**output:**

ProductID ProductName SupplierID CategoryID Unit Price

51 Manjimup Dried Apples 24 7 50 - 300 g pkgs. 53

59 Raclette Courdavault 28 4 5 kg pkg. 55