# **SQL Tutorial**

SQL tutorial provides basic and advanced concepts of SQL. Our SQL tutorial is designed for beginners and professionals.

**SQL** (Structured Query Language) is used to perform operations on the records stored in the database such as updating records, deleting records, creating and modifying tables, views, etc.

SQL is just a query language; it is not a database. To perform SQL queries, you need to install any database, for example, Oracle, MySQL, MongoDB, PostGre SQL, SQL Server, DB2, etc.

## **What is SQL**

* SQL stands for **Structured Query Language**.
* It is designed for managing data in a relational database management system (RDBMS).
* It is pronounced as S-Q-L or sometime **See-Qwell**.
* SQL is a database language, it is used for database creation, deletion, fetching rows, and modifying rows, etc.
* SQL is based on relational algebra and tuple relational calculus.

All [DBMS](https://www.javatpoint.com/dbms-tutorial) like [MySQL](https://www.javatpoint.com/mysql-tutorial), [Oracle](https://www.javatpoint.com/oracle-tutorial), MS Access, Sybase, Informix, [PostgreSQL](https://www.javatpoint.com/postgresql-tutorial), and [SQL Server](https://www.javatpoint.com/sql-server-tutorial) use SQL as standard database language.

**Why SQL is required**

SQL is required:

* To create new databases, tables and views
* To insert records in a database
* To update records in a database
* To delete records from a database
* To retrieve data from a database

## **What SQL does**

* With SQL, we can query our database in several ways, using English-like statements.
* With SQL, a user can access data from a relational database management system.
* It allows the user to describe the data.
* It allows the user to define the data in the database and manipulate it when needed.
* It allows the user to create and drop database and table.
* It allows the user to create a view, stored procedure, function in a database.
* It allows the user to set permission on tables, procedures, and views.

## **SQL statement**

SQL statements are started with any of the SQL commands/keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP etc. and the statement ends with a semicolon (;).

## **SQL Commands**

These are the some important SQL command:

* **SELECT:** it extracts data from a database.
* **UPDATE:** it updates data in database.
* **DELETE:** it deletes data from database.
* **CREATE TABLE:** it creates a new table.
* **ALTER TABLE:** it is used to modify the table.
* **DROP TABLE:** it deletes a table.
* **CREATE DATABASE:** it creates a new database.
* **ALTER DATABASE:** It is used to modify a database.
* **INSERT INTO:** it inserts new data into a database.
* **CREATE INDEX:** it is used to create an index (search key).
* **DROP INDEX:** it deletes an index.

# **SQL Data Types**

Data types are used to represent the nature of the data that can be stored in the database table. For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

Data types mainly classified into three categories for every database.

* String Data types
* Numeric Data types
* Date and time Data types

### **Data Types in MySQL, SQL Server and Oracle Databases**

### **MySQL Data Types**

A list of data types used in MySQL database. This is based on MySQL 8.0.

### **SQL Server Data Types**

**SQL Server String Data Type**

|  |  |
| --- | --- |
| **char(n)** | It is a fixed width character string data type. Its size can be up to 8000 characters. |
| **varchar(n)** | It is a variable width character string data type. Its size can be up to 8000 characters. |
| **varchar(max)** | It is a variable width character string data types. Its size can be up to 1,073,741,824  characters. |
| **text** | It is a variable width character string data type. Its size can be up to 2GB of text data. |
| **nchar** | It is a fixed width Unicode string data type. Its size can be up to 4000 characters. |
| **nvarchar** | It is a variable width Unicode string data type. Its size can be up to 4000 characters. |
| **ntext** | It is a variable width Unicode string data type. Its size can be up to 2GB of text data. |
| **binary(n)** | It is a fixed width Binary string data type. Its size can be up to 8000 bytes. |
| **varbinary** | It is a variable width Binary string data type. Its size can be up to 8000 bytes. |
| **image** | It is also a variable width Binary string data type. Its size can be up to 2GB. |

**SQL Server Numeric Data Types**

|  |  |
| --- | --- |
| **bit** | It is an integer that can be 0, 1 or null. |
| **tinyint** | It allows whole numbers from 0 to 255. |
| **Smallint** | It allows whole numbers between -32,768 and 32,767. |
| **Int** | It allows whole numbers between -2,147,483,648 and 2,147,483,647. |
| **bigint** | It allows whole numbers between -9,223,372,036,854,775,808 and  9,223,372,036,854,775,807. |
| **float(n)** | It is used to specify floating precision number data from -1.79E+308 to 1.79E+308.  The n parameter indicates whether the field should hold the 4 or 8 bytes. Default value of n  is 53. |
| **real** | It is a floating precision number data from -3.40E+38 to 3.40E+38. |
| **money** | It is used to specify monetary data from -922,337,233,685,477.5808 to  922,337,203,685,477.5807. |

**SQL Server Date and Time Data Type**

|  |  |
| --- | --- |
| **datetime** | It is used to specify date and time combination. It supports range from  January 1, 1753, to December 31, 9999 with an accuracy of 3.33 milliseconds. |
| **datetime2** | It is used to specify date and time combination. It supports range from January 1, 0001  to December 31, 9999 with an accuracy of 100 nanoseconds |
| **date** | It is used to store date only. It supports range from January 1, 0001 to  December 31, 9999 |
| **time** | It stores time only to an accuracy of 100 nanoseconds |
| **timestamp** | It stores a unique number when a new row gets created or modified. The time stamp  value is based upon an internal clock and does not correspond to real time.  Each table may contain only one-time stamp variable. |

**SQL Server Other Data Types**

|  |  |
| --- | --- |
| **Sql\_variant** | It is used for various data types except for text, timestamp, and text.  It stores up to 8000 bytes of data. |
| **XML** | It stores XML formatted data. Maximum 2GB. |
| **cursor** | It stores a reference to a cursor used for database operations. |
| **table** | It stores result set for later processing. |
| **uniqueidentifier** | It stores GUID (Globally unique identifier). |

### **Oracle Data Types**

### **Oracle String data types**

|  |  |
| --- | --- |
| **CHAR(size)** | It is used to store character data within the predefined length. It can be stored up  to 2000 bytes. |
| **NCHAR(size)** | It is used to store national character data within the predefined length. It can  be stored up to 2000 bytes. |
| **VARCHAR2(size)** | It is used to store variable string data within the predefined length. It can  be stored up to 4000 byte. |
| **VARCHAR(SIZE)** | It is the same as VARCHAR2(size). You can also use VARCHAR(size), but it  is suggested to use VARCHAR2(size) |
| **NVARCHAR2(size)** | It is used to store Unicode string data within the predefined length.  We have to must specify the size of NVARCHAR2 data type.  It can be stored up to 4000 bytes. |

**Oracle Numeric Data Types**

|  |  |
| --- | --- |
| **NUMBER(p, s)** | It contains precision p and scale s. The precision p can range from 1 to  38, and the scale s can range from -84 to 127. |
| **FLOAT(p)** | It is a subtype of the NUMBER data type. The precision p can range from 1 to 126. |
| **BINARY\_FLOAT** | It is used for binary precision(32-bit). It requires 5 bytes, including length byte. |
| **BINARY\_DOUBLE** | It is used for double binary precision (64-bit). It requires 9 bytes, including  length byte. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SQL Operators** SQL statements generally contain some reserved words or characters that are used to perform operations such as comparison and arithmetical operations etc. These reserved words or characters are known as operators.  Generally there are three types of operators in SQL:   1. SQL Arithmetic Operators 2. SQL Comparison Operators 3. SQL Logical Operators  **SQL Arithmetic Operators:** Let's assume two variables "a" and "b". Here "a" is valued 50 and "b" valued 100.  **Example:**   |  |  |  | | --- | --- | --- | | **Operators** | **Descriptions** | **Examples** | | + | It is used to add containing values of both operands | a+b will give 150 | | - | It subtracts right hand operand from left hand operand | a-b will give -50 | | \* | It multiply both operand's values | a\*b will give 5000 | | / | It divides left hand operand by right hand operand | b/a will give 2 | | % | It divides left hand operand by right hand operand and returns reminder | b%a will give 0 |  **SQL Comparison Operators:** Let's take two variables "a" and "b" that are valued 50 and 100.   |  |  |  | | --- | --- | --- | | **Operator** | **Description** | **Example** | | = | Examine both operands value that are equal or not,  if yes condition become true. | (a=b) is true | | != | This is used to check the value of both operands equal or not,  if not condition become true. | (a!=b) is true | | < > | Examines the operand's value equal or not,  if values are not equal condition is true | (a<>b) is true | | > | Examine the left operand value is greater than right Operand,  if yes condition becomes true | (a>b) is true | | < | Examines the left operand value is less than right Operand,  if yes condition becomes true | (a<b) is true | | >= | Examines that the value of left operand is greater than or equal to the value of right  operand or not, if yes condition become true | (a>=b) is true | | <= | Examines that the value of left operand is less than or equal to the value of right operand  or not, if yes condition becomes true | (a<=b) is true | | !< | Examines that the left operand value is not less than the right operand value | (a!<b) is true | | !> | Examines that the value of left operand is not greater than the value of right operand | (a!>b) is true |  **SQL Logical Operators:** This is the list of logical operators used in SQL.   |  |  | | --- | --- | | **Operator** | **Description** | | ALL | this is used to compare a value to all values in another value set. | | AND | this operator allows the existence of multiple conditions in an SQL statement. | | ANY | this operator is used to compare the value in list according to the condition. | | BETWEEN | this operator is used to search for values, that are within a set of values | | IN | this operator is used to compare a value to that specified list value | | NOT | the NOT operator reverse the meaning of any logical operator | | OR | this operator is used to combine multiple conditions in SQL statements | | EXISTS | the EXISTS operator is used to search for the presence of a row in a specified table | | LIKE | this operator is used to compare a value to similar values using wildcard operator | |

# **Syntaxes to use database :**

Create Database Database-name; 🡪 creates a database

Use Database Database-name; 🡪 uses the database

Drop Database Database-name; 🡪 Deletes the database

Rename Database old-database-name to new-database-name; 🡪renames the database name.

# **SQL Table**

Table is a collection of data, organized in terms of rows and columns. In DBMS term, table is known as relation and row as tuple.

#### **Note: A table has a specified number of columns, but can have any number of rows**

Table is the simple form of data storage. A table is also considered as a convenient representation of relations.

# **Syntaxes to use Tables :**

1. **create** **table** "tablename"  ("column1" "data type", "column2" "data type", "column3"

 "data type", …."columnN" "data type");

🡪 creates a table with the given columns and their datatypes.

Example :

**CREATE** **TABLE** Employee ( EmployeeID **int**,  FirstName **varchar**(255),

LastName **varchar**(255),  Email **varchar**(255),  AddressLine **varchar**(255),

 City **varchar**(255));

Creates a table in the database as employee with the given attributes.

=============

2.Drop table table\_name; 🡪 will delete the table

================

3.Delete from table\_name; 🡪 will delete all the data in the table

======================

4.Truncate table table\_name; 🡪  it is used to delete all the rows from the table **and free the containing space.**

When you use the drop statement it deletes the table's row together with the table's definition so all the relationships of that table with other tables will no longer be valid.

**When you drop a table:**

* Table structure will be dropped
* Relationship will be dropped
* Integrity constraints will be dropped
* Access privileges will also be dropped

On the other hand when we **TRUNCATE** a table, the table structure remains the same, so you will not face any of the above problems.

 The rollback process is not possible after truncate table statement. Once you truncate a table you cannot use a flashback table statement to retrieve the content of the table.

========================

5.Alter table table\_name Rename to new\_table\_name; 🡪will rename the table name

RENAME old\_table \_name **To** new\_table\_name; 🡪 this syntax can also be used

========================

**6.SELECT** \* **INTO** <destination\_table> **FROM** <source\_table>  ;

🡪 this will create a table with same data.

DML COMMANDS(Data Manipulation Language):

There are a total of 3 commands in dml

They are :

1.Insert

2.Delete

3.Update

Insert : this command is used to insert values into a table

Syntax : insert into table\_name values(val1,val2,val3…);

we can also insert values for specific columns

🡪insert into table\_name( col1,col2,col3..) values(val1,val2,val3….);

===================

Delete : this command is used to delete values from the table

Syntax : Delete from table\_name; 🡪 deletes all the data from table

Delete from table\_name where condition;🡪deletes the data according to the condition.

Update : update can be done in 3 different ways

1. column name update :

Update <tablename> set <colname>=<new colname>;

2.single record update :

Update <tablename> set <colname>=<value> where <condition>;

3.multiple record update :

Update <tablename> set <colname>=<value> where <condition> in <specific row id’s>;

Type of Keys :

Primary Key :

* Primary key enforces the entity integrity of the table.
* Primary key always has unique data.
* A primary key length cannot be exceeded than 900 bytes.
* A primary key cannot have null value.
* There can be no duplicate value for a primary key.
* A table can contain only one primary key constraint.

==========================

Foreign key: a foreign key is a field or a column that is used to establish a link between two tables.

In simple words you can say that, a foreign key in one table used to point primary key in another table.

==========================

Composite key:

A composite key is a combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness is guaranteed, but when it taken individually it does not guarantee uniqueness.

Sometimes more than one attributes are needed to uniquely identify an entity. A primary key that is made by the combination of more than one attribute is known as a composite key.

==========================

Unique key:

A unique key is a set of one or more than one fields/columns of a table that uniquely identify a record in a database table.

You can say that it is little like primary key but it can accept only one null value and it cannot have duplicate values.

The unique key and primary key both provide a guarantee for uniqueness for a column or a set of columns.

There is an automatically defined unique key constraint within a primary key constraint.

There may be many unique key constraints for one table, but only one PRIMARY KEY constraint for one table.

## **SQL JOIN**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

## **Different Types of SQL JOINs**

Here are the different types of the JOINs in SQL:

* **(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

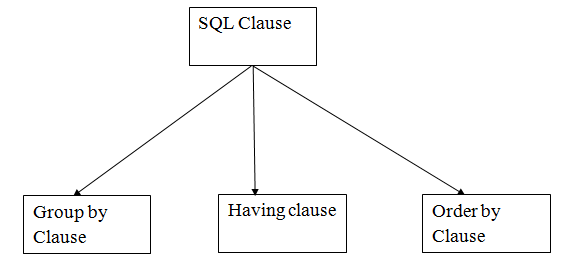
      

Example Query for a join :

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

# **SQL Clauses**

The following are the various SQL clauses:



## **1. GROUP BY**

* SQL GROUP BY statement is used to arrange identical data into groups. The GROUP BY statement is used with the SQL SELECT statement.
* The GROUP BY statement follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
* The GROUP BY statement is used with aggregation function.

**Syntax**

1. SELECT column FROM table\_name WHERE conditions  GROUP BY column ORDER BY

column

## **2. HAVING**

* HAVING clause is used to specify a search condition for a group or an aggregate.
* Having is used in a GROUP BY clause. If you are not using GROUP BY clause then you can use HAVING function like a WHERE clause.

**Syntax:**

SELECT column1, column2 FROM table\_name WHERE conditions GROUP BY column1,

column2 HAVING conditions ORDER BY column1, column2;

## **3. ORDER BY**

* The ORDER BY clause sorts the result-set in ascending or descending order.
* It sorts the records in ascending order by default. DESC keyword is used to sort the records in descending order.

**Syntax:**

SELECT column1, column2 FROM table\_name WHERE condition ORDER BY column1,

column2... ASC|DESC;

**Where**

**ASC:** It is used to sort the result set in ascending order by expression.

**DESC:** It sorts the result set in descending order by expression.