SQL - Syntax

Table of content

What is SQL Syntax?

SQL syntax is a unique set of rules and guidelines to be followed while writing SQL statements. This tutorial gives you a quick start with SQL by listing all the basic SQL Syntax.

All the SQL statements start with any of the keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW and all the statements end with a semicolon (;).

Case Sensitivity

The most important point to be noted here is that SQL is case insensitive, which means **SELECT** and **Select** have same meaning in SQL statements. Whereas, MySQL makes difference in table names. So, if you are working with MySQL, then you need to give table names as they exist in the database.

SQL Table

Let us consider a table with the name CUSTOMERS shown below, and use it as a reference to demonstrate all the SQL Statements on the same.

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	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00

SQL Statements

This tutorial lists down various SQL statements. Most of them are compatible with MySQL, Oracle, Postgres and SQL Server databases.

All the SQL statements require a **semicolon (;)** at the end of each statement. Semicolon is the standard way to separate different SQL statements which allows to include multiple SQL statements in a single line.

> *All the SQL Statements given in this tutorial have been tested with a MySQL server on Linux and Windows.*

SQL CREATE DATABASE Statement

To store data within a database, you first need to create it. This is

necessary to individualize the data belonging to an organization.

You can create a database using the following syntax –

```
CREATE DATABASE database_name;
```

Let us try to create a sample database **sampleDB** in SQL using the **CREATE DATABASE** statement –

```
CREATE DATABASE sampleDB
```

SQL USE Statement

Once the database is created, it needs to be used in order to start storing the data accordingly. Following is the syntax to change the current location to required database –

```
USE database_name;
```

We can set the previously created sampleDB as the default database by using the **USE** statement in SQL –

```
USE sampleDB;
```

SQL DROP DATABASE Statement

If a database is no longer necessary, you can also delete it. To delete/drop a database, use the following syntax –

```
DROP DATABASE database_name;
```

You can also drop the sampleDB database by using the **DROP DATABASE** statement in SQL –

```
DROP DATABASE sampleDB;
```

SQL CREATE TABLE Statement

In an SQL driven database, the data is stored in a structured manner, i.e. in the form of tables. To create a table, following syntax is used –

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);
```

The following code block is an example, which creates a CUSTOMERS table given above, 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 –

```
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)
);
```

SQL DESC Statement

Every table in a database has a structure of its own. To display the structure of database tables, we use the DESC statements. Following is the syntax –

```
DESC table_name;
```

The DESC Statement, however, only works in few RDBMS systems; hence, let us see an example by using DESC statement in the MySQL server –

```
DESC CUSTOMERS;
```

SQL INSERT INTO Statement

The SQL INSERT INTO Statement is used to insert data into database tables. Following is the syntax –

```
INSERT INTO table_name( column1, column2....columnN)
VALUES ( value1, value2....valueN);
```

The following example statements would create seven records in the empty CUSTOMERS table.

```
INSERT INTO CUSTOMERS VALUES
(1, 'Ramesh', 32, 'Ahmedabad', 2000.00 ),
(2, 'Khilan', 25, 'Delhi', 1500),
(3, 'kaushik', 23, 'Kota', 2000),
(4, 'Chaitali', 25, 'Mumbai', 6500),
(5, 'Hardik', 27, 'Bhopal', 8500),
(6, 'Komal', 22, 'Hyderabad', 4500),
(7, 'Muffy', 24, 'Indore', 10000);
```

SQL SELECT Statement

In order to retrieve the result-sets of the stored data from a database table, we use the SELECT statement. Following is the syntax –

```
SELECT column1, column2....columnN FROM table_name;
```

To retrieve the data from CUSTOMERS table, we use the SELECT statement as shown below.

```
SELECT * FROM CUSTOMERS;
```

SQL UPDATE Statement

When the stored data in a database table is outdated and needs to be updated without having to delete the table, we use the UPDATE statement. Following is the syntax –

```
UPDATE table_name
SET column1 = value1, column2 = value2....columnN=valueN
```

```
[ WHERE  CONDITION ];
```

To see an example, the following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
UPDATE CUSTOMERS SET ADDRESS = 'Pune' WHERE ID = 6;
```

SQL DELETE Statement

Without deleting the entire table from the database, you can also delete a certain part of the data by applying conditions. This is done using the DELETE FROM statement. Following is the syntax –

```
DELETE FROM table_name WHERE  {CONDITION};
```

The following code has a query, which will DELETE a customer, whose ID is 6.

```
DELETE FROM CUSTOMERS WHERE ID = 6;
```

SQL DROP TABLE Statement

To delete a table entirely from a database when it is no longer needed, following syntax is used –

```
DROP TABLE table_name;
```

This query will drop the CUSTOMERS table from the database.

SQL TRUNCATE TABLE Statement

The TRUNCATE TABLE statement is implemented in SQL to delete the data of the table but not the table itself. When this SQL statement is used, the table stays in the database like an empty table. Following is the syntax –

```
TRUNCATE TABLE table_name;
```

Following query delete all the records of the CUSTOMERS table –

```
TRUNCATE TABLE CUSTOMERS;
```

SQL ALTER TABLE Statement

The ALTER TABLE statement is used to alter the structure of a table. For instance, you can add, drop, and modify the data of a column using this statement. Following is the syntax –

```
ALTER TABLE table_name
{ADD|DROP|MODIFY} column_name {data_type};
```

Following is the example to ADD a New Column to the CUSTOMERS table using ALTER TABLE command –

```
ALTER TABLE CUSTOMERS ADD SEX char(1);
```

SQL ALTER TABLE Statement (Rename)

The ALTER TABLE statement is also used to change the name of a table as well. Use the syntax below –

```
ALTER TABLE table_name RENAME TO new_table_name;
```

Following is the example to RENAME the CUSTOMERS table using ALTER TABLE command –

```
ALTER TABLE CUSTOMERS RENAME TO NEW_CUSTOMERS;
```

SQL DISTINCT Clause

The DISTINCT clause in a database is used to identify the non-duplicate data from a column. Using the SELECT DISTINCT statement, you can retrieve distinct values from a column. Following is the syntax –

```
SELECT DISTINCT column1, column2....columnN FROM   table_
```

As an example, let us use the **DISTINCT** keyword with a SELECT query. The repetitive salary 2000.00 will only be retrieved once and the other record is ignored.

```
SELECT DISTINCT SALARY FROM CUSTOMERS ORDER BY SALARY;
```

SQL WHERE Clause

The WHERE clause is used to filter rows from a table by applying a condition. Following is the syntax to retrieve filtered rows from a

table –

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION;
```

The following query is an example to fetch all the records from CUSTOMERS table where the salary is greater than 2000, using the SELECT statement –

```
SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000;
```

SQL AND/OR Operators

The AND/OR Operators are used to apply multiple conditions in the WHERE clause. Following is the syntax –

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION-1 {AND|OR} CONDITION-2;
```

The following query is an example to fetch all the records from CUSTOMERS table where the salary is greater than 2000 AND age is less than 25, using the SELECT statement –

```
SELECT ID, NAME, SALARY FROM CUSTOMERS WHERE SALARY > 200(
```

SQL IN Clause

The IN Operator is used to check whether the data is present in the column or not, using the WHERE clause. Following is the syntax –

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name IN (val-1, val-2,...val-N);
```

For an example, we want to display records with NAME equal to 'Khilan', 'Hardik' and 'Muffy' (string values) using **IN** operator as follows –

```
SELECT * FROM CUSTOMERS
WHERE NAME IN ('Khilan', 'Hardik', 'Muffy');
```

SQL BETWEEN Clause

The BETWEEN Operator is used to retrieve the values from a table that fall in a certain range, using the WHERE clause. Following is the syntax –

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name BETWEEN val-1 AND val-2;
```

Let us try to the BETWEEN operator to retrieve CUSTOMERS records whose AGE is between 20 and 25.

```
SELECT * FROM CUSTOMERS WHERE AGE BETWEEN 20 AND 25;
```

SQL LIKE Clause

The LIKE Operator is used to retrieve the values from a table that match a certain pattern, using the WHERE clause. Following is the syntax –

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name LIKE { PATTERN };
```

As an example, let us try to display all the records from the CUSTOMERS table, where the SALARY starts with 200.

```
SELECT * FROM CUSTOMERS WHERE SALARY LIKE '200%';
```

SQL ORDER BY Clause

The ORDER BY Clause is used to arrange the column values in a given/specified order. Following is the syntax –

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

In the following example we are trying to sort the result in an ascending order by the alphabetical order of customer names –

```
SELECT * FROM CUSTOMERS ORDER BY NAME ASC;
```

SQL GROUP BY Clause

The GROUP BY Clause is used to group the values of a column together. Following is the syntax –

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name;
```

We are trying to group the customers by their age and calculate the average salary for each age group using the following query –

```
SELECT ADDRESS, AGE, SUM(SALARY)
AS TOTAL_SALARY FROM CUSTOMERS
GROUP BY ADDRESS, AGE;
```

SQL COUNT Function

The COUNT Function gives the number of non-null values present in the specified column. Following is the syntax –

```
SELECT COUNT(column_name)
FROM   table_name
WHERE  CONDITION;
```

Let us see an example –

```
SELECT AGE, COUNT(Name) FROM CUSTOMERS GROUP BY AGE;
```

SQL HAVING Clause

The HAVING clause is also used to filter a group of rows by applying a condition. Following is the syntax –

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name
HAVING (arithematic function condition);
```

In the following example, we are trying to retrieve all records from the CUSTOMERS table where the sum of their salary is greater than 5000 –

```
SELECT ADDRESS, AGE, SUM(SALARY) AS
TOTAL_SALARY FROM CUSTOMERS GROUP BY
ADDRESS, AGE HAVING TOTAL_SALARY >=5000
ORDER BY TOTAL_SALARY DESC;
```

SQL CREATE INDEX Statement

To create an index on a database table, SQL provides the CREATE INDEX statement. Following is the syntax –

```
CREATE UNIQUE INDEX index_name
ON table_name ( column1, column2,...columnN);
```

Let us create an index for the column named 'NAME' in the existing CUSTOMERS table using the following query –

```
CREATE INDEX sample_index on CUSTOMERS(NAME);
```

SQL DROP INDEX Statement

The DROP INDEX statement is used to drop an index from a table. Following is the syntax –

```
DROP INDEX index_name ON table_name;
```

Let us drop the index we created previously for the column named 'NAME' in the existing CUSTOMERS table using the following query –

```
DROP INDEX sample_index on CUSTOMERS;
```

Print Page

Advertisements