

[Trending Now](#) [DSA](#) [Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Data Science](#) [Topic-wise Practice](#) [J.](#)

# SQL | CREATE DOMAIN

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

## Used in : Postgre sql

CREATE DOMAIN creates a new domain. A domain is essentially a data type with optional constraints (restrictions on the allowed set of values). The user who defines a domain becomes its owner.

Domains are useful for abstracting common constraints on fields into a single location for maintenance. For example, several tables might contain email address columns, all requiring the same CHECK constraint to verify the address syntax. Define a domain rather than setting up each table's constraint individually.

## Examples:

```
CREATE DOMAIN CPI_DATA AS REAL CHECK
(value >= 0 AND value <= 10);
```

Now CPI\_DATA domain is create so, we can use this domain anywhere in any table of database as below :

AD

```
CREATE TABLE student(
sid char(9) PRIMARY KEY,
name varchar(30),
cpi CPI_DATA
);
```

Every time cpi\_data will check the constraint, when you add data in student table.

## Example 1 :

```
Insert into student values (201501408,Raj,7.5);
```

This will not violate the property of cpi.

### Example 2 :

```
Insert into student values (201501188,Dhaval,12);
```

ERROR. This will violate the property of cpi.

This article is contributed by **Dhaval Kumar Prajapati**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](https://www.geeksforgeeks.org/contribute) or mail your article to [contribute@geeksforgeeks.org](mailto:contribute@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 07 Sep, 2018

16

## Similar Reads

1. Extract domain of Email from table in SQL Server
2. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
3. Configure SQL Jobs in SQL Server using T-SQL
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between SQL and T-SQL
8. SQL Server | Convert tables in T-SQL into XML
9. SQL SERVER | Bulk insert data from csv file using T-SQL command
10. SQL - SELECT from Multiple Tables with MS SQL Server

Previous

Next

Article Contributed By :



# SQL CREATE TABLE

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

In the [SQL](#) database for creating a table, we use a command called **CREATE TABLE**.

## SQL CREATE TABLE Statement

A **Table** is a combination of rows and columns. For creating a table we have to define the structure of a table by adding names to columns and providing data type and size of data to be stored in columns.

### Syntax:

```
CREATE table table_name
```

AD

```
(  
  Column1 datatype (size),  
  column2 datatype (size),  
  .  
  .  
  columnN datatype(size)  
);
```

Here **table\_name** is name of the table, **column** is the name of column

## SQL CREATE TABLE Example

Let us create a table to store data of Customers, so the table name is Customer, Columns are Name, Country, age, phone, and so on.

```
CREATE TABLE Customer(  
    CustomerID INT PRIMARY KEY,  
    CustomerName VARCHAR(50),  
    LastName VARCHAR(50),  
    Country VARCHAR(50),  
    Age int(2),  
    Phone int(10)  
);
```

Output:

Customer

| CustomerID | CustomerName | LastName | Country | Age | Phone |
|------------|--------------|----------|---------|-----|-------|
| empty      |              |          |         |     |       |

## Insert Data into Table

To add data to the table, we use INSERT INTO, the syntax is as shown below:

Syntax:

*//Below query adds data in specific column, (like Column1=Value1)//*

**Insert into** Table\_name(Column1, Column2, Column3)

Values (Value1, value2, value3);

*//Below query adds data in table in sequence of column name(Value1 will be added in Column1 and so on)//*

**Insert into** Table\_name

Values (Value1, value2, value3);

*//Adding multiple data in the table in one go//*

**Insert into** Table\_name

Values (Value01, value02, value03),

(Value11, value12, value13),

(Value21, value22, value23),

(ValueN1, valueN2, valueN3)

## Example Query

This query will add data in the table named Subject

```
-- Insert some sample data into the Customers table
INSERT INTO Customer (CustomerID, CustomerName, LastName, Country, Age, Phone)
VALUES (1, 'Shubham', 'Thakur', 'India','23','xxxxxxxxxx'),
      (2, 'Aman ', 'Chopra', 'Australia','21','xxxxxxxxxx'),
      (3, 'Naveen', 'Tulasi', 'Sri lanka','24','xxxxxxxxxx'),
      (4, 'Aditya', 'Arpan', 'Austria','21','xxxxxxxxxx'),
      (5, 'Nishant. Salchichas S.A.', 'Jain', 'Spain','22','xxxxxxxxxx');
```

## Output:

Customer

| CustomerID | CustomerName                | LastName | Country   | Age | Phone      |
|------------|-----------------------------|----------|-----------|-----|------------|
| 1          | Shubham                     | Thakur   | India     | 23  | xxxxxxxxxx |
| 2          | Aman                        | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Naveen                      | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Aditya                      | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Nishant.<br>Salchichas S.A. | Jain     | Spain     | 22  | xxxxxxxxxx |

## Create a Table Using Another Table

We can also use CREATE TABLE to create a copy of an existing table. In the new table, it gets the exact column definition all columns or specific columns can be selected.

If an existing table was used to create a new table, by default the new table would be populated with the existing values from the old table.

## Syntax:

```
CREATE TABLE new_table_name AS
```

```
SELECT column1, column2,...
```

```
FROM existing_table_name
```

```
WHERE ....;
```

**Query:**

```
CREATE TABLE SubTable AS
SELECT CustomerID, CustomerName
FROM customer;
```

**Output:****SubTable**

| CustomerID | CustomerName |
|------------|--------------|
| 1          | Shubham      |
| 2          | Aman         |
| 3          | Naveen       |
| 4          | Aditya       |

This article is contributed by **Saylli Walve**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://www.geeksforgeeks.org/write-a-guest-contribution/) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 05 Apr, 2023

178

## Similar Reads

1. How to Select All Records from One Table That Do Not Exist in Another Table in SQL?
2. SQL Query to Filter a Table using Another Table
3. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
4. Configure SQL Jobs in SQL Server using T-SQL
5. SQL vs NO SQL vs NEW SQL
6. SQL | Create Table Extension
7. How to Create a Table With a Foreign Key in SQL?
8. SQL Query to Create Table With a Primary Key
9. How to Create a Table With Multiple Foreign Keys in SQL?



# SQL | DESCRIBE Statement

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

**Prerequisite:** [SQL Create Clause](#)

As the name suggests, DESCRIBE is used to describe something. Since in a database, we have tables, that's why do we use **DESCRIBE** or **DESC**(both are the same) commands to describe the **structure** of a table.

## Syntax:

```
DESCRIBE one;
```

AD

OR

```
DESC one;
```

**Note:** We can use either **DESCRIBE** or **DESC**(both are **Case Insensitive**). Suppose our table whose name is **one** has 4 columns named id, name, email, and age and all are of can **contain** null values.

## Query:

```
CREATE TABLE users (  
    id INT PRIMARY KEY,  
    name VARCHAR(50),  
    email VARCHAR(100),  
    age INT  
);
```

```
DESC users;
```

Output:

| Field | Type         | Null | Key | Default | Extra |
|-------|--------------|------|-----|---------|-------|
| id    | int(11)      | NO   | PRI | NULL    |       |
| name  | varchar(50)  | YES  |     | NULL    |       |
| email | varchar(100) | YES  |     | NULL    |       |
| age   | int(11)      | YES  |     | NULL    |       |

Here, above on using **DESC** or either **DESCRIBE** we are able to see the **structure** of a table but **not** on the console tab, the structure of the table is shown in the **describe tab** of the Database System Software.

So **desc** or **describe** command shows the **structure** of the table which include the **name** of the column, the **data type** of the column and the **nullability** which means, that column can contain null values or not.

All of these features of the table are described at the time of **Creation** of the table.

## Creating a Table or Defining the Structure of a Table

Query:

```
create table one
(
  id int not null,
  name char(25)
)
```

Here, we created a table whose name is **one** and its columns are **ID**, **NAME** and the **id** is of **not null** type i.e., we **can't** put null values in the **ID** column but we **can** put null values in the **NAME** column.

## Demonstrate DESC

**Step 1:** Defining the structure of the table.

**Creating a table:**

```
create table one
(
  id int not null,
  name char(25),
```



```
city varchar2(25)
)
```

**Step 2:** Displaying the structure of the table:

**Table:**

```
DESC one
OR
DESCRIBE one
```

**Output:**

| Field | Type     | Null | Key | Default | Extra |
|-------|----------|------|-----|---------|-------|
| id    | int(11)  | NO   |     | NULL    |       |
| name  | char(25) | YES  |     | NULL    |       |
| city  | varchar2 | YES  |     | NULL    |       |

**Note:** Here above **ID** column is of **not null** type and rest 2 columns can contain null values.

**Note:** You have to execute the DESC command on your system software only, because this command won't run on any editor. Make sure to run this command on your own installed Database **only** **References:** [Oracle.com](https://www.oracle.com)

This article is contributed by **Rajat Rawat 4**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://write.geeksforgeeks.org) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 10 May, 2023

20

## Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL INSERT INTO Statement
4. SQL | DELETE Statement
5. SQL | UPDATE Statement



# SQL | ALTER (RENAME)

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

Sometimes we may want to rename our table to give it a more relevant name. For this purpose, we can use **ALTER TABLE** to rename the name of the table. SQL ALTER TABLE is a command used to modify the structure of an existing table in a database.

**Note:** Syntax may vary in different databases.

**Syntax(Oracle,MySQL,MariaDB):**

```
ALTER TABLE table_name
```

```
RENAME TO new_table_name;
```

AD

Columns can also be given a new name with the use of **ALTER TABLE**.

**Syntax(MySQL, Oracle):**

```
ALTER TABLE table_name
```

```
RENAME COLUMN old_name TO new_name;
```

**Syntax(MariaDB):**

```
ALTER TABLE table_name
```

*CHANGE COLUMN old\_name TO new\_name;*

### Query:

```
CREATE TABLE Student (  
    id INT PRIMARY KEY,  
    name VARCHAR(50),  
    age INT,  
    email VARCHAR(50),  
    phone VARCHAR(20)  
);
```

Let's insert some data and then perform ALTER operation to understand better about alter command.

### INSERT:

```
INSERT INTO Student (id, name, age, email, phone)  
VALUES  
(1, 'Amit', 20, 'amit@gmail.com', '9999999999'),  
(2, 'Rahul', 22, 'rahul@yahoo.com', '8888888888'),  
(3, 'Priya', 21, 'priya@hotmail.com', '7777777777'),  
(4, 'Sonia', 23, 'sonia@gmail.com', '6666666666'),  
(5, 'Kiran', 19, 'kiran@yahoo.com', '5555555555');
```

### Output:

| id | name    | age | email             | phone      |
|----|---------|-----|-------------------|------------|
| 1  | Shubham | 23  | shubham@gmail.com | 9999999999 |
| 2  | Bhavika | 21  | bhavika@yahoo.com | 8888888888 |
| 3  | Aman    | 21  | aman@hotmail.com  | 7777777777 |
| 4  | Sonia   | 23  | sonia@gmail.com   | 6666666666 |
| 5  | Kiran   | 19  | kiran@yahoo.com   | 5555555555 |

### Example 1:

Change the name of column name to FIRST\_NAME in table Student.

### Syntax:

*ALTER TABLE Student RENAME COLUMN NAME TO FIRST\_NAME;*

**Query:**

```
ALTER TABLE Student RENAME name TO FIRST_NAME;
```

**Output:**

| id | FIRST_NAME | age | email             | phone      |
|----|------------|-----|-------------------|------------|
| 1  | Shubham    | 23  | shubham@gmail.com | 9999999999 |
| 2  | Bhavika    | 21  | bhavika@yahoo.com | 8888888888 |
| 3  | Aman       | 21  | aman@hotmail.com  | 7777777777 |
| 4  | Sonia      | 23  | sonia@gmail.com   | 6666666666 |
| 5  | Kiran      | 19  | kiran@yahoo.com   | 5555555555 |

Change the name of the table Student to Student\_Details.

**Query:**

```
ALTER TABLE Student RENAME TO Student_Details;
```

**Output:****Student\_Details****Student\_Details**

| id | FIRST_NAME | age | email             | phone      |
|----|------------|-----|-------------------|------------|
| 1  | Shubham    | 23  | shubham@gmail.com | 9999999999 |
| 2  | Bhavika    | 21  | bhavika@yahoo.com | 8888888888 |
| 3  | Aman       | 21  | aman@hotmail.com  | 7777777777 |
| 4  | Sonia      | 23  | sonia@gmail.com   | 6666666666 |
| 5  | Kiran      | 19  | kiran@yahoo.com   | 5555555555 |

## To Add a New Column with ALTER TABLE

To add a new column to the existing table, we first need to select the table with ALTER TABLE command table\_name, and then we will write the name of the new column and its datatype with ADD column\_name datatype. Let's have a look below to understand better.

**Syntax:**

```
ALTER TABLE table_name
```

```
ADD column_name datatype;
```

**Query:**

```
ALTER TABLE Student ADD marks INT;
```

**Output:**

| id | FIRST_NAME | age | email             | phone      | marks |
|----|------------|-----|-------------------|------------|-------|
| 1  | Shubham    | 23  | shubham@gmail.com | 9999999999 |       |
| 2  | Bhavika    | 21  | bhavika@yahoo.com | 8888888888 |       |
| 3  | Aman       | 21  | aman@hotmail.com  | 7777777777 |       |
| 4  | Sonia      | 23  | sonia@gmail.com   | 6666666666 |       |
| 5  | Kiran      | 19  | kiran@yahoo.com   | 5555555555 |       |

This article is contributed by **Shubham Chaudhary**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://write.geeksforgeeks.org) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 04 May, 2023

70

## Similar Reads

1. SQL ALTER TABLE - ADD, DROP, MODIFY
2. Difference between ALTER and UPDATE Command in SQL
3. Create, Alter and Drop schema in MS SQL Server
4. Alter login in SQL Server
5. ALTER SCHEMA in SQL Server
6. SQL Query to Drop Foreign Key Constraint Using ALTER Command
7. SQL Query to Drop Unique Key Constraints Using ALTER Command
8. SQL Query to Add Foreign Key Constraints Using ALTER Command
9. SQL Query to Add Unique key Constraints Using ALTER Command



# SQL ALTER TABLE – ADD, DROP, MODIFY

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

The **ALTER TABLE statement in SQL** is used to add, remove, or modify columns in an existing table. The ALTER TABLE statement is also used to add and remove various constraints on existing tables.

## ALTER TABLE ADD Column Statement in SQL

ADD is used to add columns to the existing table. Sometimes we may require to add additional information, in that case, we do not require to create the whole database again, **ADD** comes to our rescue.

### ALTER TABLE ADD Column Statement Syntax:

```
ALTER TABLE table_name ADD (Columnname_1 datatype,  
Columnname_2 datatype, ...Columnname_n datatype);
```

AD

The following SQL adds an “Email” column to the “Students” table:

### ALTER TABLE ADD Column Statement Example:

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

## ALTER TABLE DROP Column Statement

DROP COLUMN is used to drop columns in a table. Deleting the unwanted columns from the table.

### ALTER TABLE DROP Column Statement Syntax:

```
ALTER TABLE table_name
```

```
DROP COLUMN column_name;
```

The following SQL drop an “Email” column to the “Students” table:

### ALTER TABLE DROP Column Statement Example:

```
ALTER TABLE Students  
DROP COLUMN Email;
```

## ALTER TABLE MODIFY Column Statement in SQL

It is used to modify the existing columns in a table. Multiple columns can also be modified at once. *\*Syntax may vary slightly in different databases.*

### ALTER TABLE MODIFY Column Statement Syntax:

```
ALTER TABLE table_name
```

```
MODIFY column_name column_type;
```

### ALTER TABLE MODIFY Column Statement Syntax(SQL Server):

```
ALTER TABLE table_name
```

```
ALTER COLUMN column_name column_type;
```

### ALTER TABLE MODIFY Column Statement Example:

```
ALTER TABLE table_name  
MODIFY COLUMN column_name datatype;
```

## SQL ALTER TABLE Queries

Suppose there is a student database:

| ROLL_NO | NAME  |
|---------|-------|
| 1       | Ram   |
| 2       | Abhi  |
| 3       | Rahul |
| 4       | Tanu  |

To ADD 2 columns AGE and COURSE to table Student.

### Query:

```
ALTER TABLE Student ADD  
(AGE number(3),COURSE varchar(40));
```

### Output:

| ROLL_NO | NAME  | AGE | COURSE |
|---------|-------|-----|--------|
| 1       | Ram   |     |        |
| 2       | Abhi  |     |        |
| 3       | Rahul |     |        |
| 4       | Tanu  |     |        |

MODIFY column COURSE in table Student.

### Query:

```
ALTER TABLE Student  
MODIFY COURSE varchar(20);
```



After running the above query the maximum size of the Course Column is reduced to 20 from 40.

DROP column COURSE in table Student.

### Query:

```
ALTER TABLE Student  
DROP COLUMN COURSE;
```

### Output:

| ROLL_NO | NAME  | AGE |
|---------|-------|-----|
| 1       | Ram   |     |
| 2       | Abhi  |     |
| 3       | Rahul |     |
| 4       | Tanu  |     |

This article is contributed by **Shubham Chaudhary**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://www.geeksforgeeks.org/write-a-better-article/) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 05 Apr, 2023

77

## Similar Reads

1. Create, Alter and Drop schema in MS SQL Server
2. SQL Query to Drop Foreign Key Constraint Using ALTER Command
3. SQL Query to Drop Unique Key Constraints Using ALTER Command
4. SQL Query to Add Foreign Key Constraints Using ALTER Command
5. SQL Query to Add Unique key Constraints Using ALTER Command
6. SQL | ALTER (RENAME)



# SQL | UPDATE Statement

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

The UPDATE statement in [SQL](#) is used to update the data of an existing table in the database. We can update single columns as well as multiple columns using the UPDATE statement as per our requirement.

In a very simple way, we can say that SQL commands(UPDATE and DELETE) are used to change the data that is already in the database. The SQL DELETE command uses a WHERE clause.

## Syntax

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

AD

*WHERE condition;*

*table\_name: name of the table*

*column1: name of first , second, third column....*

*value1: new value for first, second, third column....*

*condition: condition to select the rows for which the values of columns needs to be updated.*

## Parameter Explanation

1. **UPDATE:** Command is used to update the column value in the table.
2. **WHERE:** Specifies the condition which we want to implement on the table.

**Note:** In the above query the **SET** statement is used to set new values to the particular column and the **WHERE** clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in all the rows will be updated. So the WHERE clause is used to choose the particular rows.

Let's see the SQL update statement with examples.

### Query:

```
CREATE TABLE Customer(
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(50),
    LastName VARCHAR(50),
    Country VARCHAR(50),
    Age int(2),
    Phone int(10)
);

-- Insert some sample data into the Customers table
INSERT INTO Customer (CustomerID, CustomerName, LastName, Country, Age, Phone)
VALUES (1, 'Shubham', 'Thakur', 'India', '23', 'xxxxxxxxxx'),
      (2, 'Aman ', 'Chopra', 'Australia', '21', 'xxxxxxxxxx'),
      (3, 'Naveen', 'Tulasi', 'Sri lanka', '24', 'xxxxxxxxxx'),
      (4, 'Aditya', 'Arpan', 'Austria', '21', 'xxxxxxxxxx'),
      (5, 'Nishant. Salchichas S.A.', 'Jain', 'Spain', '22', 'xxxxxxxxxx');

Select * from Customer;
```

### Output:

**Customer**

| CustomerID | CustomerName             | LastName | Country   | Age | Phone      |
|------------|--------------------------|----------|-----------|-----|------------|
| 1          | Shubham                  | Thakur   | India     | 23  | xxxxxxxxxx |
| 2          | Aman                     | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Naveen                   | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Aditya                   | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Nishant. Salchichas S.A. | Jain     | Spain     | 22  | xxxxxxxxxx |

## Update Single Column

Update the column NAME and set the value to 'Nitin' in the rows where the Age is 22.

```
UPDATE Customer SET CustomerName  
= 'Nitin' WHERE Age = 22;
```

**Output:****Customer**

| CustomerID | CustomerName | LastName | Country   | Age | Phone      |
|------------|--------------|----------|-----------|-----|------------|
| 1          | Shubham      | Thakur   | India     | 23  | xxxxxxxxxx |
| 2          | Aman         | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Naveen       | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Aditya       | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Nitin        | Jain     | Spain     | 22  | xxxxxxxxxx |

## Updating Multiple Columns

Update the columns NAME to 'Satyam' and Country to 'USA' where CustomerID is 1.

```
UPDATE Customer SET CustomerName = 'Satyam',  
Country = 'USA' WHERE CustomerID = 1;
```

**Output:****Customer**

| CustomerID | CustomerName | LastName | Country   | Age | Phone      |
|------------|--------------|----------|-----------|-----|------------|
| 1          | Satyam       | Thakur   | USA       | 23  | xxxxxxxxxx |
| 2          | Aman         | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Naveen       | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Aditya       | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Nitin        | Jain     | Spain     | 22  | xxxxxxxxxx |

**Note:** For updating multiple columns we have used comma(,) to separate the names and values of two columns.

## Omitting WHERE Clause

If we omit the WHERE clause from the update query then all of the rows will get updated.

```
UPDATE Customer SET CustomerName = 'Shubham';
```

**Output:**

The table Customer will now look like this,

**Customer**

| CustomerID | CustomerName | LastName | Country   | Age | Phone      |
|------------|--------------|----------|-----------|-----|------------|
| 1          | Shubham      | Thakur   | USA       | 23  | xxxxxxxxxx |
| 2          | Shubham      | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Shubham      | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Shubham      | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Shubham      | Jain     | Spain     | 22  | xxxxxxxxxx |

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://www.geeksforgeeks.org/write/geeksforgeeks.org) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 05 Apr, 2023

63

## Similar Reads

1. How to Update Multiple Columns in Single Update Statement in SQL?
2. How to Update Two Tables in One Statement in SQL Server?
3. Difference between Deferred update and Immediate update
4. update conflicts with concurrent update
5. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
6. Configure SQL Jobs in SQL Server using T-SQL
7. SQL vs NO SQL vs NEW SQL
8. SQL INSERT INTO Statement
9. SQL | DELETE Statement
10. SQL | INSERT IGNORE Statement

Previous

Next



# SQL | DELETE Statement

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

Pre-requisites: [SQL Commands](#)

Existing records in a table can be deleted using the SQL DELETE Statement. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

## Syntax:

```
DELETE FROM table_name WHERE some_condition;
```

AD

**table\_name:** name of the table

## Parameter Explanation

- **Some\_condition:** condition to choose a particular record.
- DELETE FROM table\_name(means we have to delete from table).

**Note:** We can delete single as well as multiple records depending on the condition we provide in the WHERE clause. If we omit the WHERE clause then all of the records will be deleted and the table will be empty.

The sample table is as follows:

**GFG\_Employees**

| id | name    | email                    | department        |
|----|---------|--------------------------|-------------------|
| 1  | Jessie  | jessie23@gmail.com       | Development       |
| 2  | Praveen | praveen_dagger@yahoo.com | HR                |
| 3  | Bisa    | dragonBall@gmail.com     | Sales             |
| 4  | Rithvik | msvv@hotmail.com         | IT                |
| 5  | Suraj   | srjsunny@gmail.com       | Quality Assurance |
| 6  | Om      | OmShukla@yahoo.com       | IT                |
| 7  | Naruto  | uzumaki@konoha.com       | Development       |

## Deleting Single Record

Delete the rows where NAME = 'Rithvik'. This will delete only the fourth row.

**Query:**

```
DELETE FROM GFG_EMPLOYEES WHERE NAME = 'Rithvik';
```

**Output:****GFG\_Employees**

| id | name    | email                    | department        |
|----|---------|--------------------------|-------------------|
| 1  | Jessie  | jessie23@gmail.com       | Development       |
| 2  | Praveen | praveen_dagger@yahoo.com | HR                |
| 3  | Bisa    | dragonBall@gmail.com     | Sales             |
| 5  | Suraj   | srjsunny@gmail.com       | Quality Assurance |
| 6  | Om      | OmShukla@yahoo.com       | IT                |
| 7  | Naruto  | uzumaki@konoha.com       | Development       |

## Deleting Multiple Records

Delete the rows from the table GFG\_EMPLOYEES where the department is "Development". This will delete 2 rows(the first row and the seventh row).

**Query:**

```
DELETE FROM GFG_EMPLOYEES  
WHERE department = 'Development';
```

**Output:****GFG\_Employees**

| id | name    | email                    | department        |
|----|---------|--------------------------|-------------------|
| 2  | Praveen | praveen_dagger@yahoo.com | HR                |
| 3  | Bisa    | dragonBall@gmail.com     | Sales             |
| 5  | Suraj   | srjsunny@gmail.com       | Quality Assurance |
| 6  | Om      | OmShukla@yahoo.com       | IT                |

**Delete All of the Records**

There are two queries to do this as shown below,

**Query:**

```
DELETE FROM GFG_EMPLOYEES;
```

Or

```
DELETE * FROM GFG_EMPLOYEES;
```

**Output:**

All of the records in the table will be deleted, there are no records left to display. The table GFG\_EMPLOYEES will become empty!

**GFG\_Employees**

| id    | name | email | department |
|-------|------|-------|------------|
| empty |      |       |            |

**Important Note:** DELETE is a [DML](#) (Data Manipulation Language) command hence operation performed by DELETE can be rolled back or undone.

[SQL Quiz](#) This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://www.geeksforgeeks.org/write-a-blog/) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.





# SQL INSERT INTO Statement

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

The INSERT INTO statement of SQL is used to insert a new row/record in a table. There are two ways of using the SQL INSERT INTO statement for inserting rows.

## SQL INSERT Query

### 1. Only Values

The first method is to specify only the value of data to be inserted without the column names.

#### INSERT INTO Syntax:

```
INSERT INTO table_name VALUES (value1, value2, value3);
```

*table\_name: name of the table. value1, value2*

AD

*value of first column, second column,... for the new record*

### Column Names And Values Both

In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:

#### Insert Data in Specified Columns – Syntax:

```
INSERT INTO table_name (column1, column2, column3)
```

*VALUES ( value1, value2, value3); table\_name:*

*name of the table.*

*column1: name of first column, second column .*

*value1, value2, value3 value of first column, second column,... for the new record*

Suppose there is a Student database and we want to add values.

| ROLL_NO | NAME   | ADDRESS | PHONE          | AGE |
|---------|--------|---------|----------------|-----|
| 1       | Ram    | Delhi   | xxxxxxxxxxxxxx | 18  |
| 2       | RAMESH | GURGAON | xxxxxxxxxxxxxx | 18  |
| 3       | SUJIT  | ROHTAK  | xxxxxxxxxxxxxx | 20  |
| 4       | SURESH | ROHTAK  | xxxxxxxxxxxxxx | 18  |
| 3       | SUJIT  | ROHTAK  | xxxxxxxxxxxxxx | 20  |
| 2       | RAMESH | GURGAON | xxxxxxxxxxxxxx | 18  |

### Method 1 (Inserting only values) – SQL INSERT Query

If we want to insert only values then we use the following query:

**Query:**

```
INSERT INTO Student VALUES
('5', 'HARSH', 'WEST BENGAL',
'XXXXXXXXXX', '19');
```

**Output:**

The table **Student** will now look like this:

| ROLL_NO | NAME   | ADDRESS | PHONE      | Age |
|---------|--------|---------|------------|-----|
| 1       | Ram    | Delhi   | XXXXXXXXXX | 18  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |

| ROLL_NO | NAME   | ADDRESS     | PHONE      | Age |
|---------|--------|-------------|------------|-----|
| 3       | SUJIT  | ROHTAK      | XXXXXXXXXX | 20  |
| 4       | SURESH | Delhi       | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK      | XXXXXXXXXX | 20  |
| 2       | RAMESH | GURGAON     | XXXXXXXXXX | 18  |
| 5       | HARSH  | WEST BENGAL | XXXXXXXXXX | 19  |

## Method 2 (Inserting values in only specified columns) – SQL INSERT INTO Statement

If we want to insert values in the specified columns then we use the following query:

### Query:

```
INSERT INTO Student (ROLL_NO,
NAME, Age) VALUES ('5', 'PRATIK', '19');
```

### Output:

The table **Student** will now look like this:

| ROLL_NO | NAME   | ADDRESS | PHONE      | Age |
|---------|--------|---------|------------|-----|
| 1       | Ram    | Delhi   | XXXXXXXXXX | 18  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 4       | SURESH | Delhi   | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 5       | PRATIK | null    | null       | 19  |

Notice that the columns for which the values are not provided are filled by null. Which are the default values for those columns?

## 2. Using SELECT in INSERT INTO Statement

We can use the SELECT statement with INSERT INTO statement to copy rows from one table and insert them into another table. The use of this statement is similar to that of the INSERT INTO statement. The difference is that the SELECT statement is used here to select data from a different table. The different ways of using INSERT INTO SELECT statement are shown below:

### Inserting all columns of a table – INSERT INTO SELECT Statement

We can copy all the data of a table and insert it into a different table.

#### Syntax:

```
INSERT INTO first_table SELECT * FROM second_table;
```

*first\_table: name of first table.*

*second\_table: name of second table.*

We have used the SELECT statement to copy the data from one table and the INSERT INTO statement to insert from a different table.

### Inserting specific columns of a table – INSERT INTO SELECT Statement

We can copy only those columns of a table that we want to insert into a different table.

#### Syntax:

```
INSERT INTO first_table(names_of_columns1)
```

```
SELECT names_of_columns2 FROM second_table;
```

*first\_table: name of first table. second\_table: name of second table.*

*names of columns1: name of columns separated by comma(,) for table 1.*

*names of columns2: name of columns separated by comma(,) for table 2.*

We have used the SELECT statement to copy the data of the selected columns only from the second table and the INSERT INTO statement to insert in the first table.

### Copying specific rows from a table – INSERT INTO SELECT Statement

We can copy specific rows from a table to insert into another table by using the WHERE clause with the SELECT statement. We have to provide appropriate conditions in the WHERE clause to select specific rows.

*INSERT INTO table1 SELECT \* FROM table2 WHERE condition;*

*first\_table: name of first table.*

*second\_table: name of second table.*

*condition: condition to select specific rows.*

Suppose there is a LateralStudent database.

| ROLL_NO | NAME   | ADDRESS   | PHONE      | Age |
|---------|--------|-----------|------------|-----|
| 7       | SOUVIK | HYDERABAD | XXXXXXXXXX | 18  |
| 8       | NIRAJ  | NOIDA     | XXXXXXXXXX | 19  |
| 9       | SOMESH | ROHTAK    | XXXXXXXXXX | 20  |

### Method 1 – (Inserting all rows and columns)

If we want to insert only values then we use the following query:

#### SQL INSERT INTO SELECT Query:

```
INSERT INTO Student
SELECT * FROM LateralStudent;
```

#### Output:

This query will insert all the data of the table LateralStudent in the table Student. The table Student will now look like this,

| ROLL_NO | NAME | ADDRESS | PHONE      | Age |
|---------|------|---------|------------|-----|
| 1       | Ram  | Delhi   | XXXXXXXXXX | 18  |

| ROLL_NO | NAME   | ADDRESS | PHONE      | Age |
|---------|--------|---------|------------|-----|
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 4       | SURESH | Delhi   | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 7       | SOUVIK | DUMDUM  | XXXXXXXXXX | 18  |
| 8       | NIRAJ  | NOIDA   | XXXXXXXXXX | 19  |
| 9       | SOMESH | ROHTAK  | XXXXXXXXXX | 20  |

### Method 2(Inserting specific columns)

If we want to insert values in the specified columns then we use the following query:

#### SQL INSERT INTO SELECT Query:

```
INSERT INTO Student(ROLL_NO,NAME,Age)
SELECT ROLL_NO, NAME, Age FROM LateralStudent;
```

#### Output:

This query will insert the data in the columns ROLL\_NO, NAME, and Age of the table LateralStudent in the table Student and the remaining columns in the Student table will be filled by *null* which is the default value of the remaining columns. The table Student will now look like this,

| ROLL_NO | NAME   | ADDRESS | PHONE      | Age |
|---------|--------|---------|------------|-----|
| 1       | Ram    | Delhi   | XXXXXXXXXX | 18  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |

| ROLL_NO | NAME   | ADDRESS | PHONE      | Age |
|---------|--------|---------|------------|-----|
| 4       | SURESH | Delhi   | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 7       | SOUVIK | null    | null       | 18  |
| 8       | NIRAJ  | null    | null       | 19  |
| 9       | SOMESH | null    | null       | 20  |

### Select specific rows to insert:

```
INSERT INTO Student SELECT *  
FROM LateralStudent WHERE Age = 18;
```

### Output:

This query will select only the first row from table LateralStudent to insert into the table Student. The table Student will now look like this,

| ROLL_NO | NAME   | ADDRESS | PHONE      | Age |
|---------|--------|---------|------------|-----|
| 1       | Ram    | Delhi   | XXXXXXXXXX | 18  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 4       | SURESH | Delhi   | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 7       | SOUVIK | DUMDUM  | XXXXXXXXXX | 18  |

## To insert multiple rows in a table using Single SQL Statement:

### Syntax:

```
INSERT INTO table_name(Column1,Column2,Column3,.....)
VALUES (Value1, Value2,Value3,.....),
      (Value1, Value2,Value3,.....),
      (Value1, Value2,Value3,.....),
      ..... ;
```

Where,

- **table\_name:** name of the table.  
**Column 1:** name of the first column, second column.
- **Values:** Value1, Value2, Value3: the value of the first column, second column.
- For each new row inserted, you need To provide Multiple lists of values where each list is separated by “,”. Every list of values corresponds to values to be inserted in each new row of the table. Values in the next list tell values to be inserted in the next Row of the table.

### Example:

The following SQL statement inserts multiple rows in Student Table.

### Query:

```
INSERT INTO STUDENT(ID, NAME,AGE,GRADE,CITY)
VALUES(1,"AMIT KUMAR",15,10,"DELHI"),
      (2,"GAURI RAO",18,12,"BANGALORE"),
      (3,"MANAV BHATT",17,11,"NEW DELHI"),
      (4,"RIYA KAPOOR",10,5,"UDAIPUR");
```

### Output:

Thus STUDENT Table will look like this:

| ID | NAME       | AGE | GRADE | CITY      |
|----|------------|-----|-------|-----------|
| 1  | AMIT KUMAR | 15  | 10    | DELHI     |
| 2  | GAURI RAO  | 18  | 12    | BANGALORE |



| ID | NAME        | AGE | GRADE | CITY      |
|----|-------------|-----|-------|-----------|
| 3  | MANAV BHATT | 17  | 11    | NEW DELHI |
| 4  | RIYA KAPOOR | 10  | 5     | UDAIPUR   |

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://www.geeksforgeeks.org/write-a-better-article/) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 12 Apr, 2023

64

## Similar Reads

1. Insert multiple values into multiple tables using a single statement in SQL Server
2. Insert Into Select statement in MS SQL Server
3. SQL | INSERT IGNORE Statement
4. Insert statement in MS SQL Server
5. SQL SERVER | Bulk insert data from csv file using T-SQL command
6. SELECT INTO Statement in SQL
7. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
8. Configure SQL Jobs in SQL Server using T-SQL
9. SQL vs NO SQL vs NEW SQL
10. SQL Server | Convert tables in T-SQL into XML

[Previous](#)

[Next](#)

**Article Contributed By :**



# SQL | SELECT Query

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

The SELECT Statement in SQL is used to retrieve or fetch data from a [database](#).

We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. This result table is also called the result set. With the SELECT clause of a SELECT command statement, we specify the columns that we want to be displayed in the query result and, optionally, which column headings we prefer to see above the result table.

The select clause is the first clause and is one of the last clauses of the select statement that the database server evaluates. The reason for this is that before we can determine what to include in the final result set, we need to know all of the possible columns that could be included in the final result set.

## CREATE TABLE:

AD

```
CREATE TABLE Customer(  
    CustomerID INT PRIMARY KEY,  
    CustomerName VARCHAR(50),  
    LastName VARCHAR(50),  
    Country VARCHAR(50),  
    Age int(2),  
    Phone int(10)  
);  
-- Insert some sample data into the Customers table  
INSERT INTO Customer (CustomerID, CustomerName, LastName, Country, Age, Phone)  
VALUES (1, 'Shubham', 'Thakur', 'India', '23', 'xxxxxxxxxx'),  
       (2, 'Aman ', 'Chopra', 'Australia', '21', 'xxxxxxxxxx'),  
       (3, 'Naveen', 'Tulasi', 'Sri lanka', '24', 'xxxxxxxxxx'),
```

```
(4, 'Aditya', 'Arpan', 'Austria', '21', 'xxxxxxxxxx'),  
(5, 'Nishant. Salchichas S.A.', 'Jain', 'Spain', '22', 'xxxxxxxxxx');
```

**Output:**

| CustomerID | CustomerName             | LastName | Country   | Age | Phone      |
|------------|--------------------------|----------|-----------|-----|------------|
| 1          | Shubham                  | Thakur   | India     | 23  | xxxxxxxxxx |
| 2          | Aman                     | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Naveen                   | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Aditya                   | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Nishant. Salchichas S.A. | Jain     | Spain     | 22  | xxxxxxxxxx |

To fetch any column in the table.

**Syntax:**

```
SELECT column1,column2 FROM table_name
```

*column1 , column2: names of the fields of the table*

*table\_name: from where we want to apply query*

This query will return all the rows in the table with fields column1 and column2.

## SELECT Statement in SQL

To fetch the entire table or all the fields in the table:

**Syntax:**

```
SELECT * FROM table_name;
```

*— asterisks represent all attributes of the table*

Query to fetch the fields CustomerName, LastName from the table Customer:

```
SELECT CustomerName, LastName FROM Customer;
```

**Output:**

| CustomerName             | LastName |
|--------------------------|----------|
| Shubham                  | Thakur   |
| Aman                     | Chopra   |
| Naveen                   | Tulasi   |
| Aditya                   | Arpan    |
| Nishant. Salchichas S.A. | Jain     |

To fetch all the fields from the table Customer:

```
SELECT * FROM Customer;
```

**Output:**

| CustomerID | CustomerName             | LastName | Country   | Age | Phone      |
|------------|--------------------------|----------|-----------|-----|------------|
| 1          | Shubham                  | Thakur   | India     | 23  | xxxxxxxxxx |
| 2          | Aman                     | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Naveen                   | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Aditya                   | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Nishant. Salchichas S.A. | Jain     | Spain     | 22  | xxxxxxxxxx |

## SELECT Statement with WHERE Clause

Suppose we want to see table values with specific conditions then Where Clause is used with select statement.

**Query:**

```
SELECT CustomerName FROM Customer where Age = '21';
```

**Output:**

| CustomerName |
|--------------|
| Aman         |
| Aditya       |

## SQL SELECT Statement with GROUP BY Clause

**Query:**

```
SELECT COUNT (item), Customer_id FROM Orders GROUP BY order_id;
```

**Output:**

| COUNT (item) | customer_id |
|--------------|-------------|
| 1            | 4           |
| 1            | 4           |
| 1            | 3           |
| 1            | 1           |
| 1            | 2           |

## SELECT Statement with HAVING Clause

Consider the following database for Having Clause:

**Results** Messages

|   | EmployeeId ▾ | Name ▾ | Gender ▾ | Salary ▾ | Department ▾ | Experience ▾ |
|---|--------------|--------|----------|----------|--------------|--------------|
| 1 | 1            | Rachit | M        | 50000    | Engineering  | 6 year       |
| 2 | 2            | Shobit | M        | 37000    | HR           | 3 year       |
| 3 | 3            | Isha   | F        | 56000    | Sales        | 7 year       |
| 4 | 4            | Devi   | F        | 43000    | Management   | 4 year       |
| 5 | 5            | Akhil  | M        | 90000    | Engineering  | 15 year      |

**Query:**

```
SELECT Department, sum(Salary) as Salary  
FROM employee  
GROUP BY department  
HAVING SUM(Salary) >= 50000;
```

**Output:**

## Results Messages

|   | Department ▼ | Salary ▼ |
|---|--------------|----------|
| 1 | Engineering  | 140000   |
| 2 | Sales        | 56000    |

## SELECT Statement with ORDER BY clause in SQL

Query:

```
SELECT * FROM Customer ORDER BY Age DESC;
```

Output:

| CustomerID | CustomerName             | LastName | Country   | Age | Phone      |
|------------|--------------------------|----------|-----------|-----|------------|
| 3          | Naveen                   | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 1          | Shubham                  | Thakur   | India     | 23  | xxxxxxxxxx |
| 5          | Nishant. Salchichas S.A. | Jain     | Spain     | 22  | xxxxxxxxxx |
| 2          | Aman                     | Chopra   | Australia | 21  | xxxxxxxxxx |
| 4          | Aditya                   | Arpan    | Austria   | 21  | xxxxxxxxxx |

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://www.geeksforgeeks.org/write-article/) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 07 May, 2023

81

## Similar Reads



# SQL | DROP, TRUNCATE

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

SQL commands are broadly classified into two types [DDL](#), [DML](#) here we will be learning about DDL commands, and in DDL we will be learning about DROP and TRUNCATE in this article.

## DROP

DROP is used to delete a whole [database](#) or just a table.

In this article, we will be learning about the DROP statement which destroys objects like an existing database, table, index, or view. A DROP statement in SQL removes a component from a relational database management system (RDBMS).

### Syntax:

AD

*DROP object object\_name*

### Examples 1:

#### To Drop a table

```
DROP TABLE table_name;
```

**table\_name:** Name of the table to be deleted.

### Examples 2:

#### To Drop a database

```
DROP DATABASE database_name;
```

**database\_name:** Name of the database to be deleted.

## TRUNCATE

The major difference between TRUNCATE and DROP is that truncate is used to delete the data inside the table not the whole table.

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extent of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing several integrity-enforcing mechanisms. It was officially introduced in the SQL:2008 standard. The TRUNCATE TABLE mytable statement is logically (though not physically) equivalent to the DELETE FROM mytable statement (without a WHERE clause).

### Syntax:

```
TRUNCATE TABLE table_name;
```

*table\_name:* Name of the table to be truncated.

*DATABASE name – student\_data*

### DROP vs TRUNCATE

- Truncate is normally ultra-fast and it's ideal for deleting data from a temporary table.
- Truncate preserves the structure of the table for future use, unlike drop table where the table is deleted with its full structure.
- Table or Database deletion using a DROP statement **cannot** be rolled back, so it must be used wisely.

### Difference between DROP and TRUNCATE

| DROP   | TRUNCATE  |
|--|---|
| In the drop table data and its definition is deleted with their full structure.                                | It preserves the structure of the table for further use exist but deletes all the data. |
| Drop is used to eliminate existing complications and fewer complications in the whole database from the table. | Truncate is used to eliminate the tuples from the table.                                |



| DROP  | TRUNCATE  |
|---|---|
| Integrity constraints get removed in the DROP command.  | Integrity constraint doesn't get removed in the Truncate command.                 |
| Since the structure does not exist, the View of the table does not exist in the Drop command. | Since the structure exists, the View of the table exists in the Truncate command. |
| Drop query frees the table space complications from memory.                                   | This query does not free the table space from memory.                             |
| It is slow as there are so many complications compared to the TRUNCATE command.               | It is fast as compared to the DROP command as there are fewer complications.      |

let's consider the given database:

| Student |        |         |            |     |
|---------|--------|---------|------------|-----|
| ROLL_NO | NAME   | ADDRESS | PHONE      | Age |
| 1       | Ram    | Delhi   | XXXXXXXXXX | 18  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 4       | SURESH | Delhi   | XXXXXXXXXX | 18  |
| 3       | SUJIT  | ROHTAK  | XXXXXXXXXX | 20  |
| 2       | RAMESH | GURGAON | XXXXXXXXXX | 18  |

To delete the whole database

**Query:**

```
DROP DATABASE student_data;
```

After running the above query whole database will be deleted.

To truncate the Student\_details table from the student\_data database.

**Query:**

```
TRUNCATE TABLE Student_details;
```

After running the above query Student\_details table will be truncated, i.e, the data will be deleted but the structure will remain in the memory for further operations.

This article is contributed by **Pratik Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://write.geeksforgeeks.org) or mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 12 Jun, 2023

78

## Similar Reads

1. Difference between DROP and TRUNCATE in SQL
2. Difference between DELETE, DROP and TRUNCATE
3. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
4. Configure SQL Jobs in SQL Server using T-SQL
5. SQL vs NO SQL vs NEW SQL
6. SQL ALTER TABLE - ADD, DROP, MODIFY
7. Difference between DELETE and DROP in SQL
8. Create, Alter and Drop schema in MS SQL Server
9. Drop login in SQL Server
10. CREATE and DROP INDEX Statement in SQL

[Previous](#)

[Next](#)

Article Contributed By :



GeeksforGeeks

Vote for difficulty

Current difficulty : [Easy](#)



# SQL | INSERT IGNORE Statement

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

We know that a primary key of a table cannot be duplicated. For instance, the roll number of a student in the student table must always be distinct. Similarly, the EmployeeID is expected to be unique in an employee table. When we try to insert a tuple into a table where the primary key is repeated, it results in an error. However, with the INSERT IGNORE statement, we can prevent such errors from popping up, especially when inserting entries in bulk and such errors can interrupt the flow of insertion. Instead, only a warning is generated.

## Cases where INSERT IGNORE avoids error

- Upon insertion of a duplicate key where the column must contain a PRIMARY KEY or UNIQUE constraint
- Upon insertion of NULL value where the column has a NOT NULL constraint.
- Upon insertion of a row to a partitioned table where the inserted values go against the partition format.

## Example:

Say we have a relation, Employee.

AD

## Employee Table:

| EmployeeID | Name   | City      |
|------------|--------|-----------|
| 15001      | Aakash | Delhi     |
| 15003      | Sahil  | Bangalore |

|       |         |           |
|-------|---------|-----------|
| 15010 | John    | Hyderabad |
| 15008 | Shelley | Delhi     |
| 15002 | Ananya  | Mumbai    |
| 15004 | Sia     | Pune      |

As we can notice, the entries are not sorted on the basis of their primary key, i.e. EmployeeID.

### Sample Query:

```
INSERT IGNORE INTO Employee (EmployeeID, Name, City)
VALUES (15002, 'Ram', 'Mumbai');
```

### Output:

No entry inserted.

### Sample Query:

#### Inserting multiple records

When inserting multiple records at once, any that cannot be inserting will not be, but any that can will be:

```
INSERT IGNORE INTO Employee (EmployeeID, Name, City)
VALUES (15007, 'Shikha', 'Delhi'), (15002, 'Ram', 'Mumbai'), (15009, 'Sam',
'A Ahmedabad');
```

### Output:

The first and the last entries get inserted; the middle entry is simple ignored. No error is flashed.

### Disadvantage

Most users do not prefer INSERT IGNORE over INSERT since some errors may slip unnoticed. This may cause inconsistencies in the table, thereby causing some tuples to not get inserted without the user having a chance to correct them. Hence, INSERT IGNORE must be used in very specific conditions.

This article is contributed by **Anannya Uberoi**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](https://www.geeksforgeeks.org/contribute) or mail your article to [contribute@geeksforgeeks.org](mailto:contribute@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.



# SQL | Case Statement

[Read](#)[Discuss](#)[Courses](#)[Practice](#)

Control statements form the heart of most languages since they control the execution of other sets of statements. These are also found in [SQL](#) and should be exploited for uses such as query filtering and query optimization by carefully selecting tuples that match our requirements.

In this article, we explore the Case-Switch statement in SQL. The CASE statement is SQL's way of handling if/then logic.

There can be two valid ways of going about the case-switch statements.

The first takes a variable called `case_value` and matches it with some `statement_list`.

AD

## Syntax:

```
CASE case_value  
  
WHEN when_value THEN statement_list  
  
[WHEN when_value THEN statement_list] ...  
  
[ELSE statement_list]  
  
END CASE
```

The second considers a `search_condition` instead of variable equality and executes the `statement_list` accordingly.

## Syntax:

*CASE*

*WHEN search\_condition THEN statement\_list*

*[WHEN search\_condition THEN statement\_list] ...*

*[ELSE statement\_list]*

*END CASE*

### Example:

#### CREATE TABLE:

Below is a selection from the “Customer” table in the sample database:

```
CREATE TABLE Customer(
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(50),
    LastName VARCHAR(50),
    Country VARCHAR(50),
    Age int(2),
    Phone int(10)
);
-- Insert some sample data into the Customers table
INSERT INTO Customer (CustomerID, CustomerName, LastName, Country, Age, Phone)
VALUES (1, 'Shubham', 'Thakur', 'India', '23', 'xxxxxxxxxx'),
       (2, 'Aman ', 'Chopra', 'Australia', '21', 'xxxxxxxxxx'),
       (3, 'Naveen', 'Tulasi', 'Sri lanka', '24', 'xxxxxxxxxx'),
       (4, 'Aditya', 'Arpan', 'Austria', '21', 'xxxxxxxxxx'),
       (5, 'Nishant. Salchichas S.A.', 'Jain', 'Spain', '22', 'xxxxxxxxxx');
```

### Output:

| CustomerID | CustomerName             | LastName | Country   | Age | Phone      |
|------------|--------------------------|----------|-----------|-----|------------|
| 1          | Shubham                  | Thakur   | India     | 23  | xxxxxxxxxx |
| 2          | Aman                     | Chopra   | Australia | 21  | xxxxxxxxxx |
| 3          | Naveen                   | Tulasi   | Sri lanka | 24  | xxxxxxxxxx |
| 4          | Aditya                   | Arpan    | Austria   | 21  | xxxxxxxxxx |
| 5          | Nishant. Salchichas S.A. | Jain     | Spain     | 22  | xxxxxxxxxx |

## Adding Multiple Conditions to a CASE statement

**Query:**

By adding multiple conditions in SQL

```
SELECT CustomerName, Age,  
CASE  
    WHEN Age > 22 THEN 'The Age is greater than 20'  
    WHEN Age = 21 THEN 'The Age is 21'  
    ELSE 'The Age is over 30'  
END AS AgeText  
FROM Customer ;
```

**Output:**

| CustomerName             | Age | QuantityText               |
|--------------------------|-----|----------------------------|
| Shubham                  | 23  | The Age is greater than 20 |
| Aman                     | 21  | The Age is 21              |
| Naveen                   | 24  | The Age is greater than 20 |
| Aditya                   | 21  | The Age is 21              |
| Nishant. Salchichas S.A. | 22  | The Age is over 30         |

## CASE Statement With ORDER BY Clause

**Query:**

By using Order by Clause in SQL

```
SELECT CustomerName, Country  
FROM Customer  
ORDER BY  
(CASE  
    WHEN Country IS 'India' THEN Country  
    ELSE Age  
END);
```

**Output:**

| CustomerName             | Country   |
|--------------------------|-----------|
| Aman                     | Australia |
| Aditya                   | Austria   |
| Nishant. Salchichas S.A. | Spain     |
| Naveen                   | Sri lanka |
| Shubham                  | India     |

## Some important points about CASE statements:

1. There should always be a SELECT in the case statement.
2. END. ELSE is an optional component but WHEN THEN these cases must be included in the CASE statement.
3. We can make any conditional statement using any conditional operator (like [WHERE](#)) between WHEN and THEN. This includes stringing together multiple conditional statements using AND and OR.
4. We can include multiple WHEN statements and an ELSE statement to counter with unaddressed conditions.

Last Updated : 13 Apr, 2023

29

## Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL INSERT INTO Statement
4. SQL | DELETE Statement
5. SQL | UPDATE Statement
6. SQL | INSERT IGNORE Statement
7. SQL | DESCRIBE Statement
8. SQL | MERGE Statement
9. MERGE Statement in SQL Explained
10. Delete statement in MS SQL Server

[Previous](#)

[Next](#)

## Article Contributed By :



GeeksforGeeks