# **PostgreSQL Practice Using SQL Shell**

# Basic of Installation

## Staring

To start SQL Shell, at first install PostgreSQL from official [website](https://www.enterprisedb.com/downloads/postgres-postgresql-downloads). Then find it in start menu or search in search bar. Next, you will find it like this terminal. Hit enter for all, but enter your password that you have set for installation.



Then you will find it like this.



## Check Version

After entering password, you will see the current version of PostgreSQL.



Or you can execute this command to see current version.

SELECT version();



## List of Databases

To see list of database, you can run this command.

\l



# Create Database

There are many ways to create database. We use simple method.

## To create DB

The basic syntax of CREATE DATABASE statement is as follows −

CREATE DATABASE dbname;

where *dbname* is the name of a database to create. Our database name is TestDB.



To see list of database, you can run this command.

\l



# Create Table

## To create table

To create table CREATE TABLE command is needed, then table name and its field name, data type.

CREATE TABLE cars (  
  brand VARCHAR(255),  
  model VARCHAR(255),  
  year INT  
);

When you execute the above statement, an empty table named cars will be created, and the SQL Shell application will return the following CREATE TABLE:



## To see all data of table

You can "display" the empty table you just created with another SQL statement:

SELECT \* FROM cars;



# Insert Data

## To insert data in table

To insert data into a table in PostgreSQL, we use the INSERT INTO statement.

The following SQL statement will insert one row of data into the cars table.

INSERT INTO cars (brand, model, year)  
VALUES ('Ford', 'Mustang', 1964);



The SQL Shell application will return the following:

INSERT 0 1

Which means that 1 row was inserted.

**Note:**

As you can see in the SQL statement above, string values must be written with apostrophes.

Numeric values can be written without apostrophes, but you can include them if you want.

To check the result we can display the table with this SQL statement:

SELECT \* FROM cars;



## Insert Multiple Rows

To insert multiple rows of data, we use the same INSERT INTO statement, but with multiple values:

INSERT INTO cars (brand, model, year)  
VALUES  
  ('Volvo', 'p1800', 1968),  
  ('BMW', 'M1', 1978),  
  ('Toyota', 'Celica', 1975);





## Select/Fetch Data

To retrieve data from a database, we use the SELECT statement.

### Specify Columns

By specifying the column names, we can choose which columns to select:

SELECT brand, year FROM cars;



### Return ALL Columns

Specify a \* instead of the column names to select all columns:

SELECT \* FROM cars;



# Add Column

## The ALTER TABLE Statement

To add a column to an existing table, we have to use the ALTER TABLE statement.

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

## Add Column

When adding columns we must also specify the data type of the column. Our color column will be a string, and we specify string types with the VARCHAR keyword. we also want to restrict the number of characters to 255:

ALTER TABLE cars  
ADD color VARCHAR(255);



The SQL Shell application will return the following:

ALTER TABLE

To check the result we can display the table with this SQL statement:

SELECT \* FROM cars;



As you can see, the cars table now has a color column. The new column is empty.

## UPDATE

The UPDATE statement is used to modify the value(s) in existing records in a table.

Set the color of the Volvo to 'red':

UPDATE cars  
SET color = 'red'  
WHERE brand = 'Volvo';



WE see that SQL Shell return UPDATE 1, which means that 1 row was affected by the UPDATE statement.

**Note:**

1. Be careful with the WHERE clause, in the example above ALL rows where brand = 'Volvo' gets updated.
2. Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

To check the result we can display the table with this SQL statement:

SELECT \* FROM cars;



## Update Multiple Columns

To update more than one column, separate the name/value pairs with a comma ,:

Update color and year for the Toyota

UPDATE cars  
SET color = 'white', year = 1970  
WHERE brand = 'Toyota';



To check the result we can display the table with this SQL statement:

SELECT \* FROM cars;



# Alter Column

## The ALTER TABLE Statement

To change the data type, or the size of a table column we have to use the ALTER TABLE statement.

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

## Alter Column

We want to change the data type of the year column of the cars table from INT to VARCAHR(4).

To modify a column, use the ALTER COLUMN statement and the TYPE keyword followed by the new data type:

ALTER TABLE cars  
ALTER COLUMN year TYPE VARCHAR(4);



SQL Shell will return ALTER TABLE.

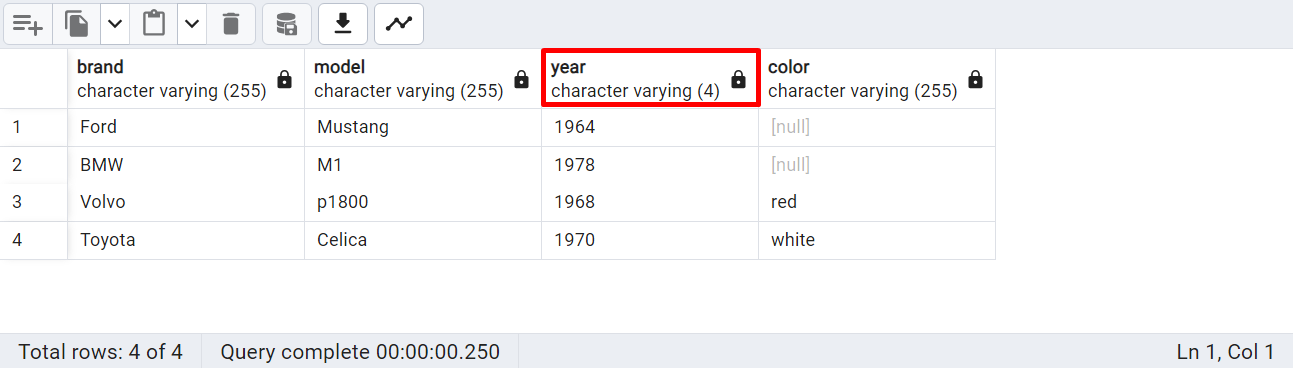
**Note:** Some data types cannot be converted if the column has value. E.g. numbers can always be converted to text, but text cannot always be converted to numbers.

You can see how many table are available in your databases. Just run below command in your SQL Shell.

\dt



You will find it in pgAdmin 4 that shows year data type is changed.



## Change Maximum Allowed Characters

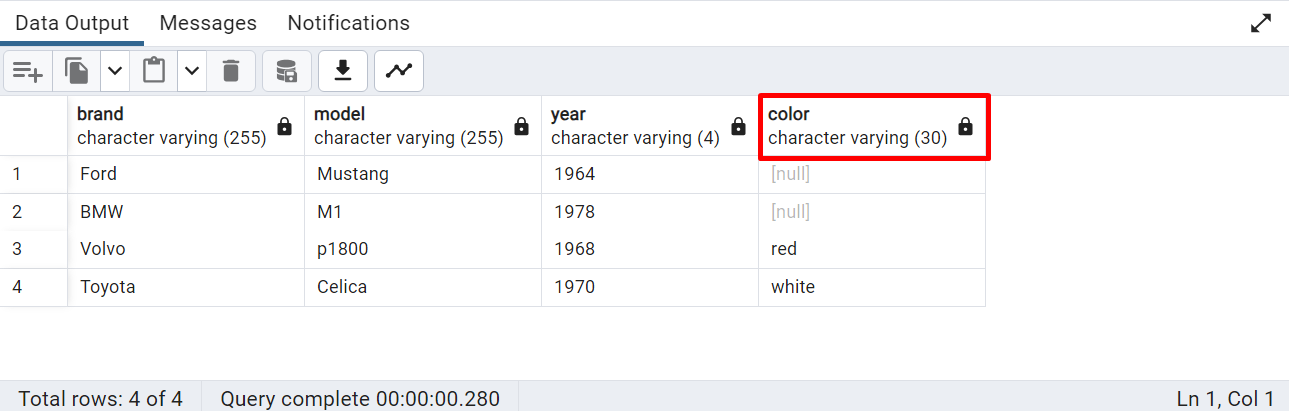
We also want to change the maximum number of characters allowed in the color column of the cars table.

Change the color column from VARCHAR(255) to VARCHAR(30):

ALTER TABLE cars  
ALTER COLUMN color TYPE VARCHAR(30);



SQL Shell also return ALTER TABLE. Your changed is available in pgAdmin 4.



# Drop Column

## The ALTER TABLE Statement

To remove a column from a table, we have to use the ALTER TABLE statement.

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

## Drop Column

We want to remove the column named color from the cars table.

To remove a column, use the DROP COLUMN statement:

ALTER TABLE cars  
DROP COLUMN color;



We will not see color column in our table anymore. Just run this command.

SELECT \* FROM cars;



# Delete

## The DELETE Statement

The DELETE statement is used to delete existing records in a table.

**Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted.

If you omit the WHERE clause, **all records in the table will be deleted**!.

To to delete the record(s) where brand is 'Volvo', use this statement:

DELETE FROM cars  
WHERE brand = 'Volvo';



DELETE 1, which means that 1 row was deleted.

If you run below command, you will see that deleted row is not shown.

SELECT \* FROM cars;



## Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact.

The following SQL statement deletes all rows in the cars table, without deleting the table:

DELETE FROM cars;

We will see DELETE 3 SQL shell, which means that 3 row was deleted.

## Truncate Table

Because we omit the WHERE clause in the DELETE statement above, all records will be deleted from the cars table.

The same would have been achieved by using the TRUNCATE TABLE statement:

Delete all records in the cars table:

TRUNCATE TABLE cars;

If we run this statement, we will see

SELECT \* FROM cars;

