# CREATE TABLE Syntax

*by Sophia Tutorial*

WHAT'S COVERED

This tutorial explores using the CREATE TABLE statement to generate tables in the database in two parts:

1. Rules and Data Types
2. Using the CREATE TABLE Command

## 1. Rules and Data Types

The rules for table and column names can be a bit different depending on the type of database, so it is important to consider looking at the documentation from the database that you intend to use. There are some common rules that are consistent across many different databases. These include:

- The table and column names must start with a letter.
- The table and column names can only contain letters, numbers, and underscores.
- The table and column names should not contain spaces (though some databases may allow this).
- The table and column names have a limited length of characters that can be used.

For best practices, spaces should not have corresponding spaces in the table or column names. Otherwise, there are some characters – depending on the database – that would need to be used to enclose the table or column name.
For each column or field in a table, we must identify the type of data that the column can store, along with its length in some cases. The most common data types are:

- Boolean: Stores true, false, or null (no value).
- CHAR(n): A fixed-length character of length n with space added. If a string is added to the column that is shorter than the fixed length, PostgreSQL pads extra spaces up to the length (n) of the column. If we try to insert a value that is longer than the length of the column, an error will be generated.
- VARCHAR(n): A variable-length character string that can store up to n characters. Note that with VARCHAR, unlike CHAR, PostgreSQL does not pad the spaces when the stored string is shorter than the length of the column.
- TEXT: A variable-length character string that has unlimited length.
- SMALLINT: A small integer is a 2-byte signed integer that has a range between -32,768 to 32,767.
- INT: An integer is a 4-byte integer that has a range between -2,147,483,648 to 2,147,483,647.

- SERIAL: The serial is the same as an integer but PostgreSQL will automatically generate and populate the values into this column. We'll cover this in an upcoming tutorial.
- Float(n): This is a floating-point number that specifies precision that can be up to 8 bytes.
- Real: A 4-byte floating-point number.
- Numeric(p,s): A real number that has p digits, with s number of digits that are displayed after the decimal point.
- Date: Stores the dates on its own.
- Time: Stores the time of day values.
- Timestamp: Stores both the date and time values.

---

# 2. Using the CREATE TABLE Command

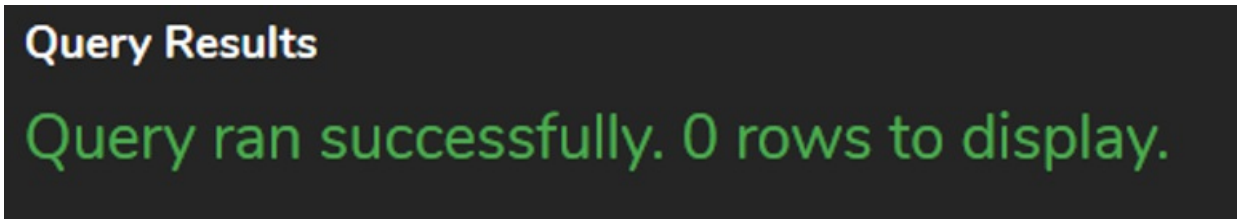
Now that we know some rules to get started, we can move to the CREATE TABLE syntax. The SQL CREATE TABLE command makes a new table by defining the layout of the table. Similar to the SELECT statement that you've learned about, there is a standard structure for the format of the CREATE TABLE statement. After the CREATE TABLE statement, we have the name of the table, and the names, data types, and lengths of the columns.

The basic syntax looks like the following:

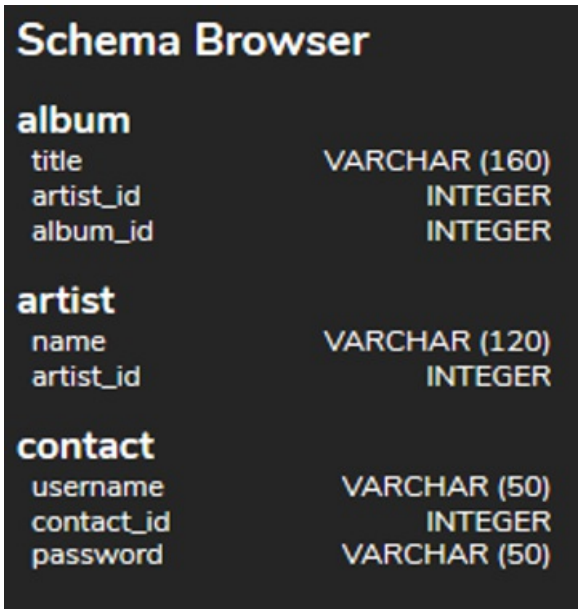
CREATE TABLE <tablename> ( <column1> <datatype> [constraint], <column2> <datatype> [constraint], ... );
In the sample above, the <tablename> and <column> would need to be replaced with the actual table name and column. The <datatype> would be replaced with the data type and size, if necessary. The [constraint] is optional, and we will get into the different options in the next tutorial. For now, the only constraint you need to be aware of is the PRIMARY KEY constraint. The PRIMARY KEY constraint uniquely identifies a row in a table. There is an open parenthesis after the <tablename> to signify the beginning of the column list and then a close parenthesis after all of the columns and constraints have been created. If the CREATE TABLE statement is missing either one, it will raise an error.

To make the code easier to read, it is good practice to have one line per column or attribute. If we wanted to create a new table called contact that had the contact_id as the primary key and two additional columns with the username and password, the statement could look like the following:

CREATE TABLE contact( contact_id int PRIMARY KEY, username VARCHAR(50), password VARCHAR(50) );
Running the command, you should see a message similar to the following:




In the Schema Browser, you should see the contact table listed after the album and artist table:




A table could have just one column or many columns, depending on what you are trying to create. For example, you could have a newsletter table that just consists of emails:

CREATE TABLE newsletter( email VARCHAR(50) );
Although this is the basic syntax, there are other criteria that we'll explore in upcoming tutorials.

---

## Video Transcription

In order to create tables in the database, you would utilize the create table statement. The create table statement looks similar to this, where it has CREATE TABLE as key words, then the table name. In our case, it's contact. Then we open up with parentheses, close up with the parentheses, and end the statement with a semicolon. Within the parentheses, you have your call names, the data types, and any constraints associated with them.

In this case here, we have three different columns. We have contact ID. That's set up as "int." It's going to be set up with the constraint as a primary key. We also have the username that's set up as a VARCHAR(50), and password as a VARCHAR(50). Each one of these is going to be separated by a comma. In a table, you can have one column up to whatever number of columns that the database can hold. In Postgres, that'll be 1,600 columns that you can actually have in a table.

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

To create a table, we use the CREATE TABLE statement, which may include the table name, column

names, data types, and data length if applicable, as well as any constraints.

Source: Authored by Vincent Tran