Creating tables

This lesson shows you how to create a new table and populate the new table with values in SQL. You'll also learn how to change values in a table or delete the table altogether.

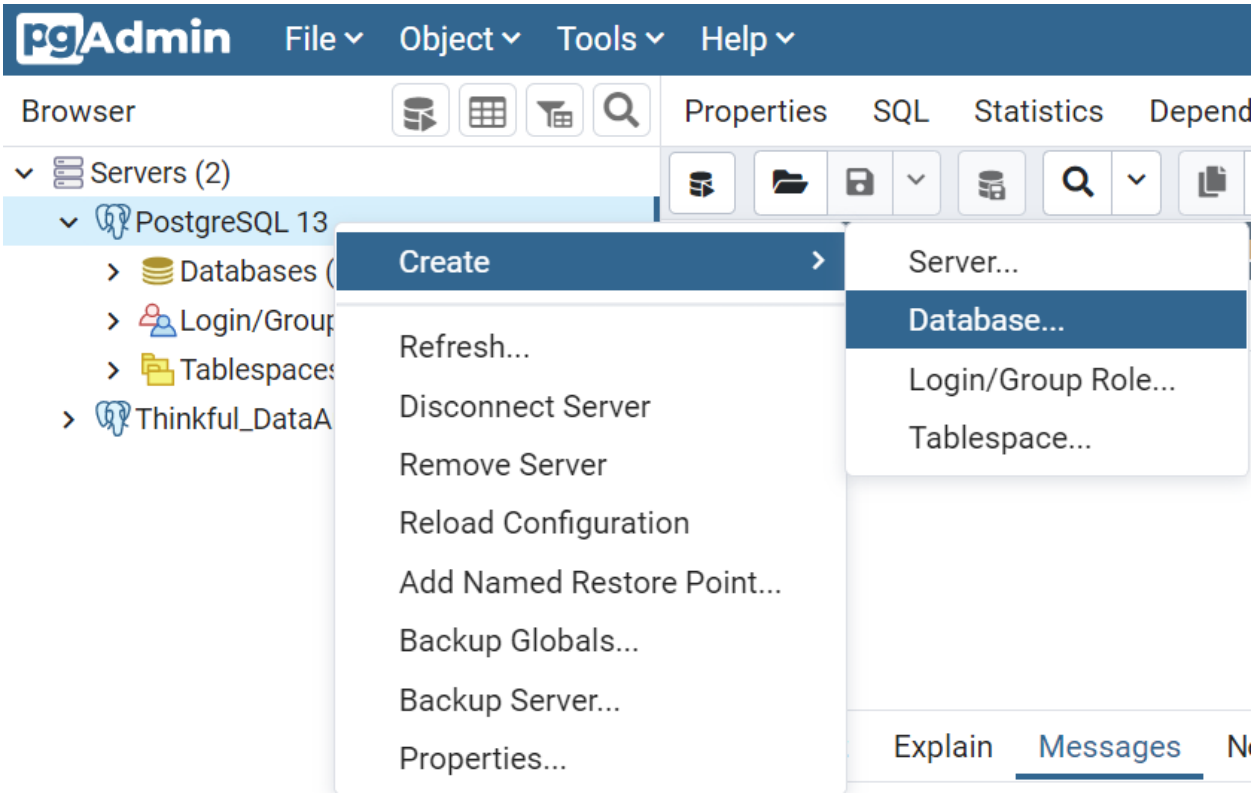
So far, you've created queries for tables that were already created on the Thinkful server. In this lesson, you'll learn how to create your very own table! This will be useful if you ever have any information or data that you want to store or be able to query from.

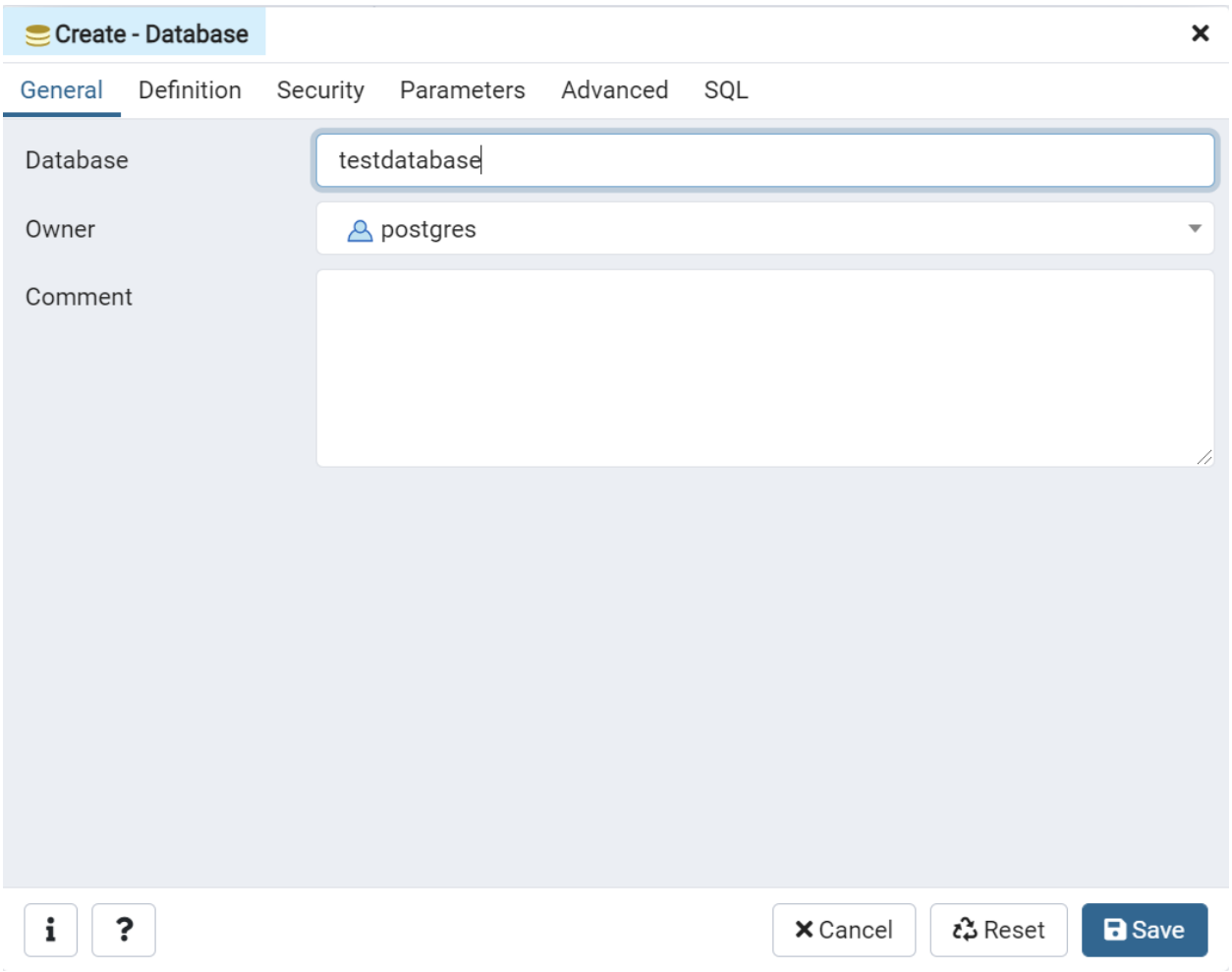
Creating a new table using CREATE TABLE

One day, you may need to create a table that's too large for Excel. In that case, you can create a SQL table and store it in PostgreSQL using pgAdmin. Although this probably isn't something that you need to do today, it's a useful skill to know, especially if you're considering a career in data analytics or data engineering.

Creating a new table is a straightforward process. However, the Thinkful_DataAnalytics server that you've been using so far doesn't allow you to create new tables. So, to create a new table, use the default server that came prepackaged in pgAdmin. This is the PostgreSQL 13 server in the tree control pane. Right-click this server and click Create > Database. Give your database whatever name you like, such as testdatabase. Then press Save.

Make sure that you're logged in to the PostgreSQL 13 server. Otherwise, you may not have the same options.





Now that you have your own database, you can begin creating tables within it. Right-click your new database in the tree control pane, and then click Query Tool.

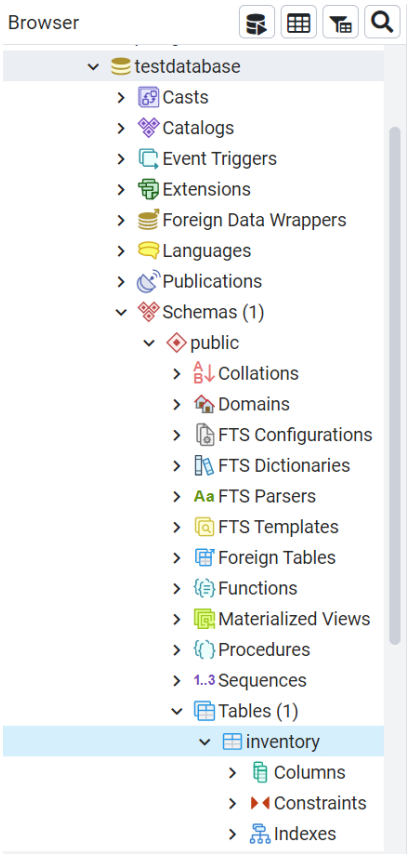
The first step of creating a new table is to create the shell of the table using CREATE TABLE. The shell of the table defines the number of columns that you want, the names of the columns, and the data types of the columns. The structure of this code is as follows:

```
CREATE TABLE table_name (
column1_name datatype,
column2_name datatype,
column3_name datatype, …
);
```

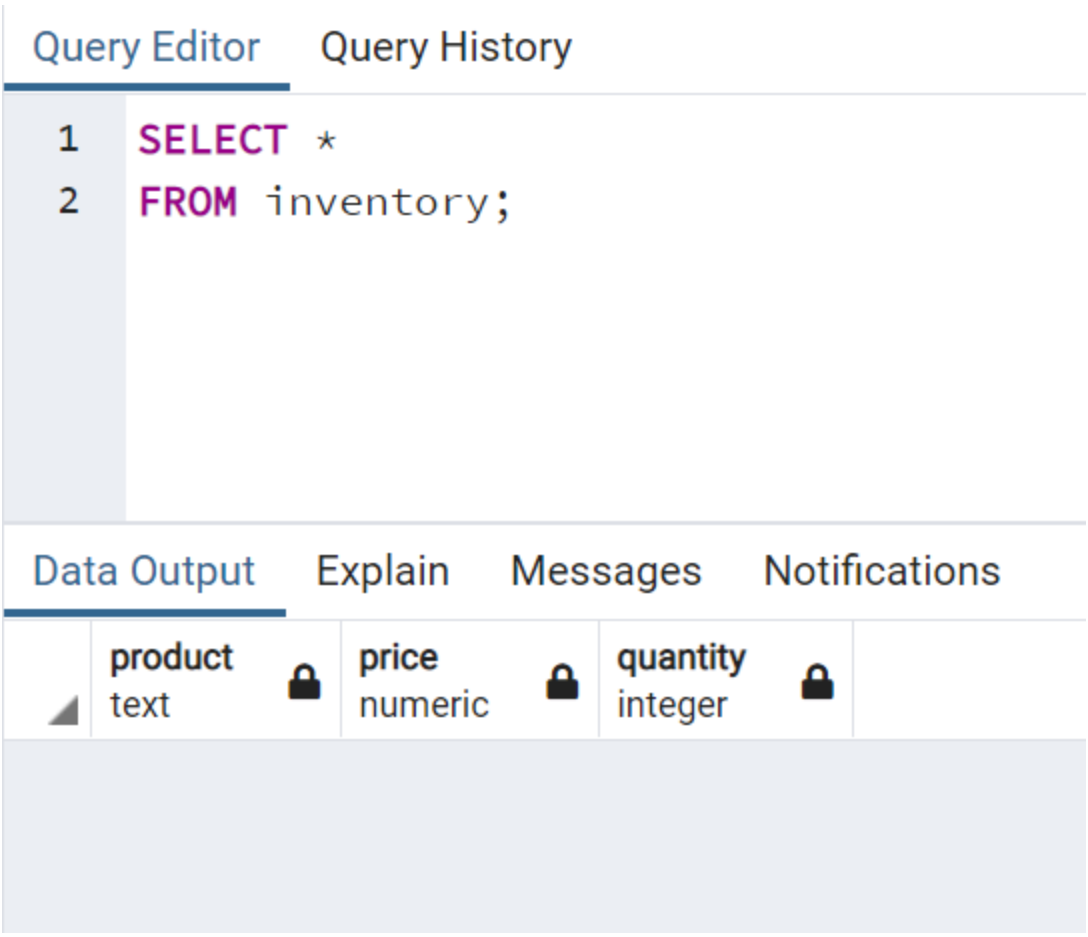
Say you want to create a table called inventory. You want this table to have three columns: a text column called product, a decimal column called price, and an integer column called quantity. The following code creates that table shell.

```
CREATE TABLE inventory (
product  text,
price decimal,
quantity integer
);
```

Create and execute this code in pgAdmin. If you run into any issues, double-check your punctuation. When your query has run successfully, go back to the tree control pane. Right-click your database and click Refresh. Now scroll down and expand Schemas. Then expand Tables. You will see your inventory table show up.



Go ahead and check out how your table shell looks so far; write and execute a SELECT * FROM query on this table. So far, only column names and data types show up in the output—which makes sense, considering you haven't added any data values yet.



Adding a row of values using INSERT INTO

To add one row of values to your new table, you can use the INSERT INTO statement. There are a few ways to use INSERT INTO; for this course, use the following structure:

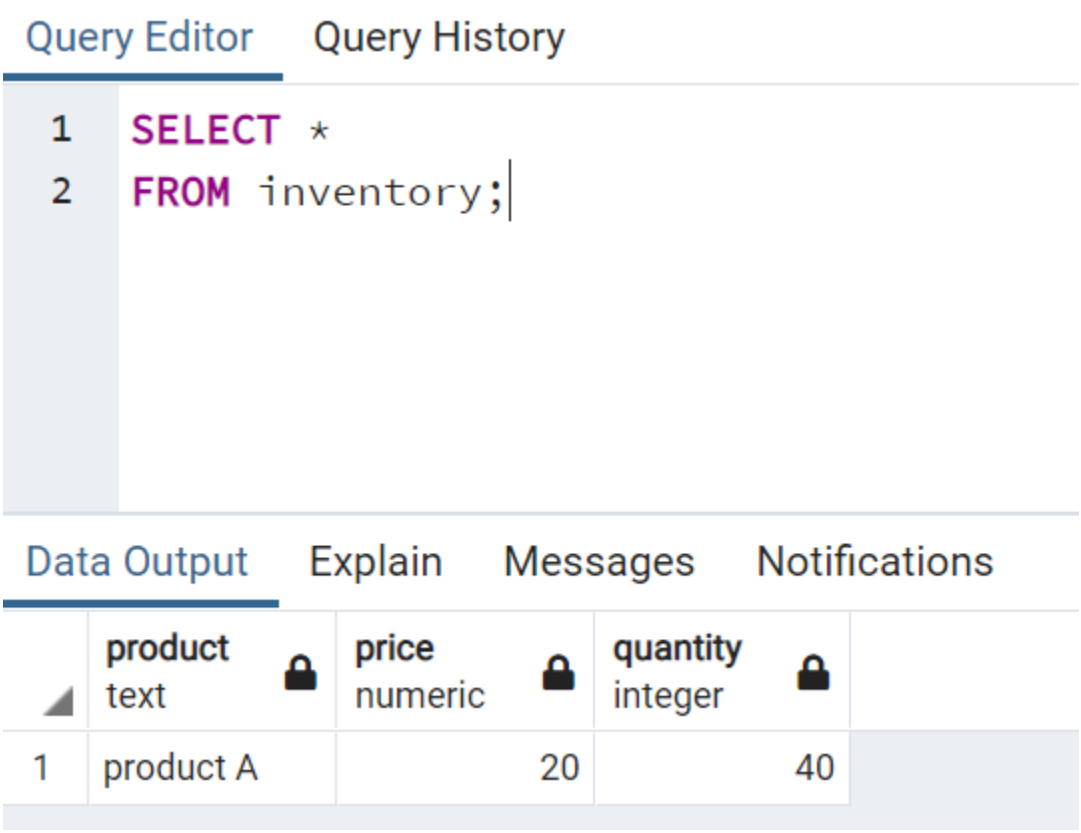
```
INSERT INTO table_name (column1_name, column2_name, column3_name, … )
VALUES (value1, value2, value3);
```

The column names specify which columns you're entering values for. The value names correspond to each of the column names.

The code below inserts one row with three values: product A for product, 20 for price, and 40 for quantity.

```
INSERT INTO inventory (product, price, quantity)
VALUES ('product A', 20, 40);
```

Execute this query in pgAdmin. Now check to see how your table looks by executing a SELECT * FROM query on your table.



Congratulations! You just created your very own table using SQL!

Adding multiple rows of values

If you want to add multiple rows of values, you can use the following structure for an INSERT INTO statement.

```
INSERT INTO table_name (column1_name, column2_name, column3_name,…)
VALUES (value1, value2, value3),
(value1, value2, value3),
(value1, value2, value3),… ;
```

Say you want to add two additional rows to your new table inventory. You want to add the following rows of data:

```
product B, 25.75, 45
product C, 22.50, 50
```

Use the following code to add these two additional rows at the same time.

```
INSERT INTO inventory (product, price, quantity)
VALUES ('product B', 25.75, 45),
('product C', 22.50, 50);
```

Execute this query and then execute a SELECT * FROM query on your inventory table to see your fancy new table.

Query Editor | Query History

```
SELECT *
FROM inventory;
```

Data Output | Explain | Messages | Notifications

	product text	price numeric	quantity integer
1	product A	20	40
2	product B	25.75	45
3	product C	22.50	50

Updating a table using UPDATE

When creating a table, you may accidentally type a value wrong. If that happens, don't worry—SQL has a clause that lets you fix this. The UPDATE clause allows you to update any value in a table. This clause has the following structure:

```
UPDATE table_name
SET column1_name = value1, column2_name = value2, column3_name = value3,…
WHERE condition;
```

For example, imagine that the quantity for product C should actually be 60, not 50. The following code corrects that.

```
1  UPDATE inventory
2  SET quantity = 60
3  WHERE product = 'product C';
```

After executing this query, execute a simple SELECT * FROM on this table to see the change.

Query Editor Query History

```
1  SELECT *
2  FROM inventory;
```

Data Output Explain Messages Notifications

	product text	price numeric	quantity integer
1	product A	20	40
2	product B	25.75	45
3	product C	22.50	60

Deleting a table using DROP TABLE

If for some reason you decide to delete the table (maybe you don't need it anymore, or you have another larger table that already includes data from this table), you can use the DROP TABLE statement. The DROP TABLE statement uses the following structure:

```
1  DROP TABLE table_name;
```

Go ahead and try this out with the table that you just made, or create another table for practice and delete that one.