

# 2

## Creating a Physical Database

In Chapter 1, we discussed entity relationships and normalization to build conceptual designs of databases. In this chapter, we will move onto the physical design and building databases in SQL Server.

### CHAPTER OBJECTIVES

In this chapter, you will understand how to do the following:

- Understand how to start SQL Server.
- Understand how to create databases.
- Understand how to create tables.
- Understand how to choose appropriate data types for columns in a table.
- Understand how to create an entity-relationship diagram.

## Starting SQL Server

You can find SQL Server in your list of programs via the start menu as shown in Figure 1. From the program menu select “**Microsoft SQL Server 2012**” (or the version year you are using). From there navigate further below and select “**SQL Server Management Studio**”, as seen in Figure 2.

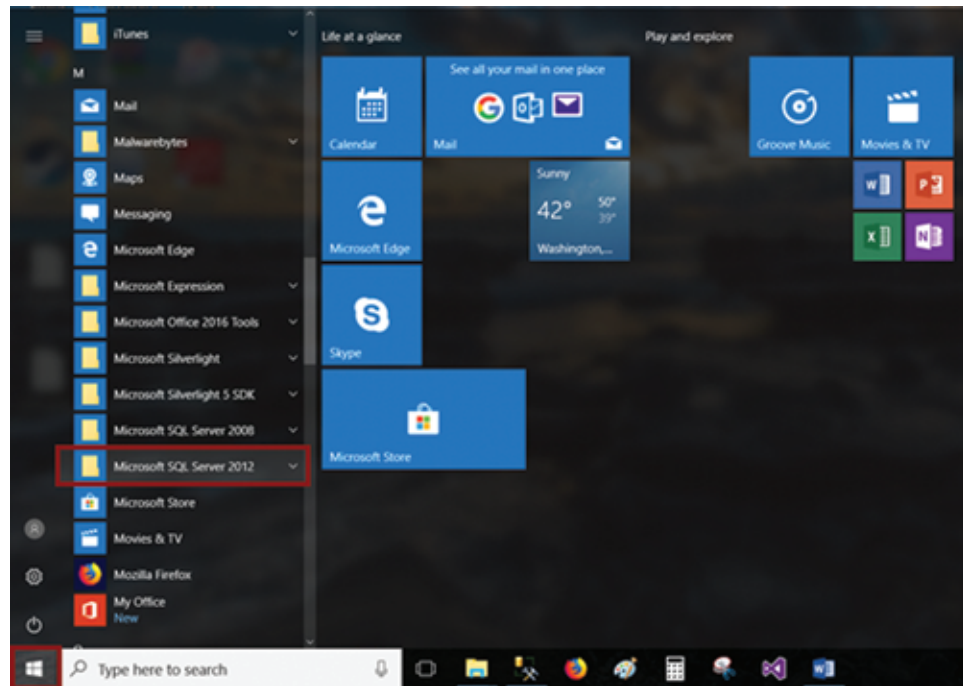


Figure 1

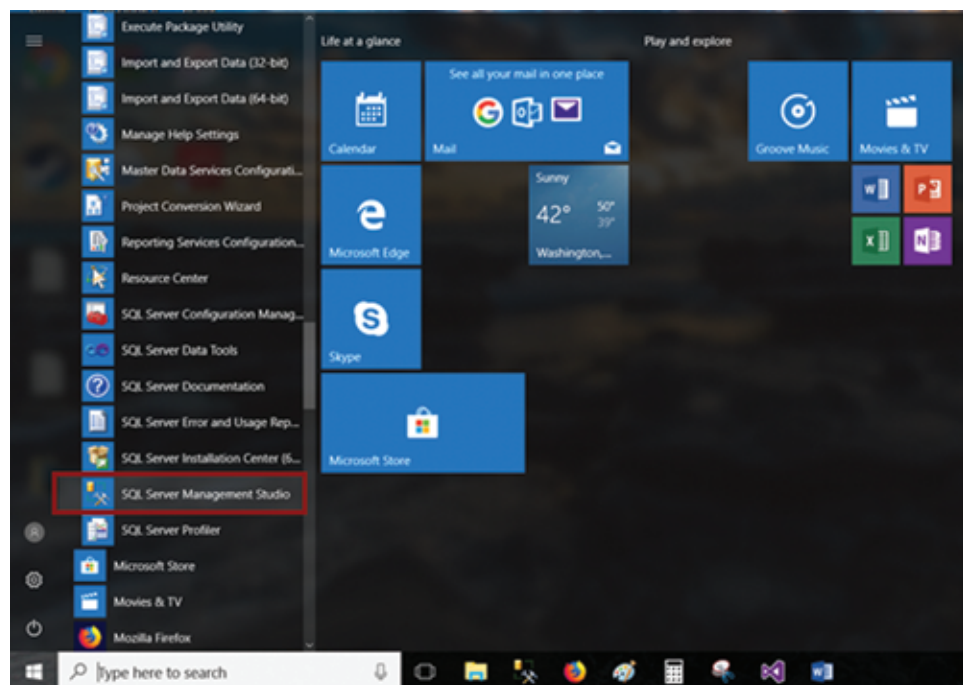


Figure 2

Finding and launching SQL can be even more user friendly by simply typing “SQL” in the Windows search bar as shown below in Figure 3.

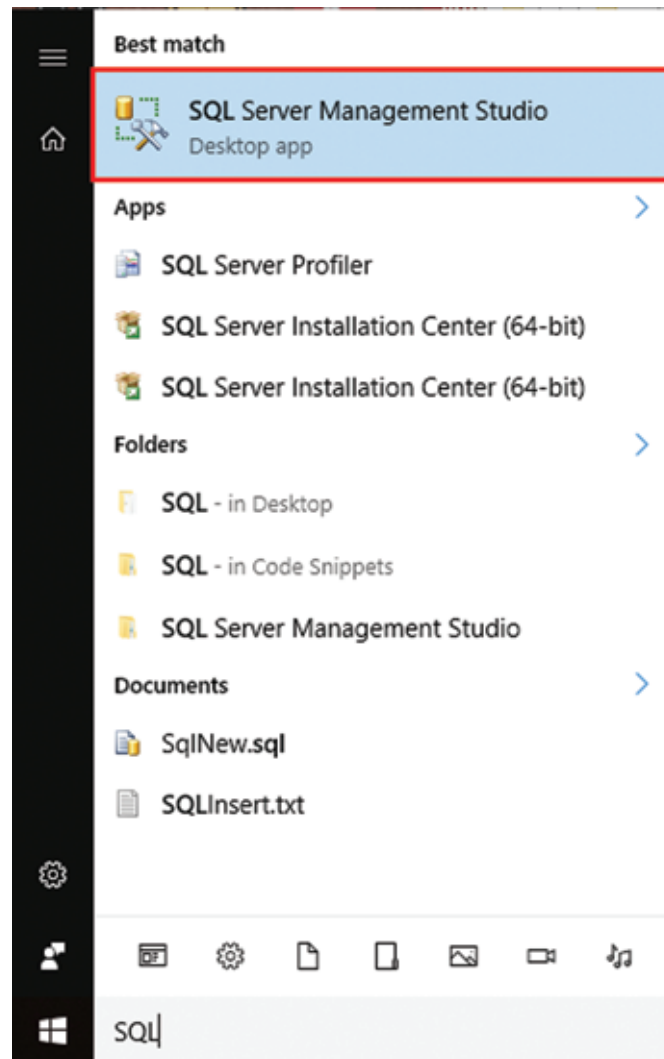


Figure 3

## Connecting to SQL Server

Figure 4 shows the “Connect to Server” dialog window. You want to make sure the following parameters are used:

1. Server type = Database Engine
2. Server name = Your computer name. Sometimes in place of your computer name it will default to “localhost” and this can work too.
3. Authentication = Windows Authentication
4. Clicking “Connect” will launch SQL Server.

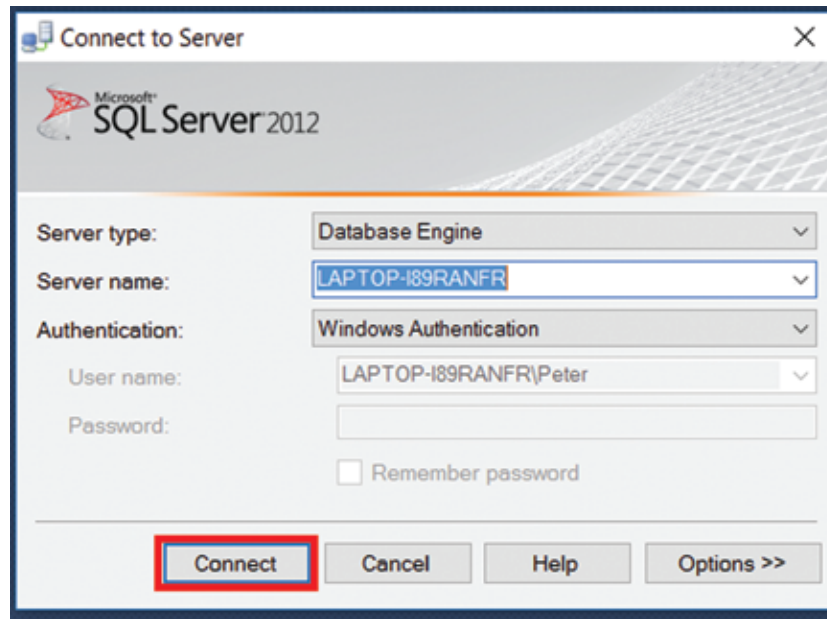


Figure 4

► **NOTE:** In case you don't know your computer name you can find it by following the directions below.

**Windows 7 :** Left click the start button > select “computer” > select “system properties” in the top menu bar > you will see a list of information that includes your computer name.

**Window 10:** Right click the Windows icon > Select System > you will see a list of information that includes your computer name.

Occasionally, the default server name chosen by SQL Server isn't the correct one. This results in the error message shown in Figure 5. This is a common error for first-time users and is easily resolved.

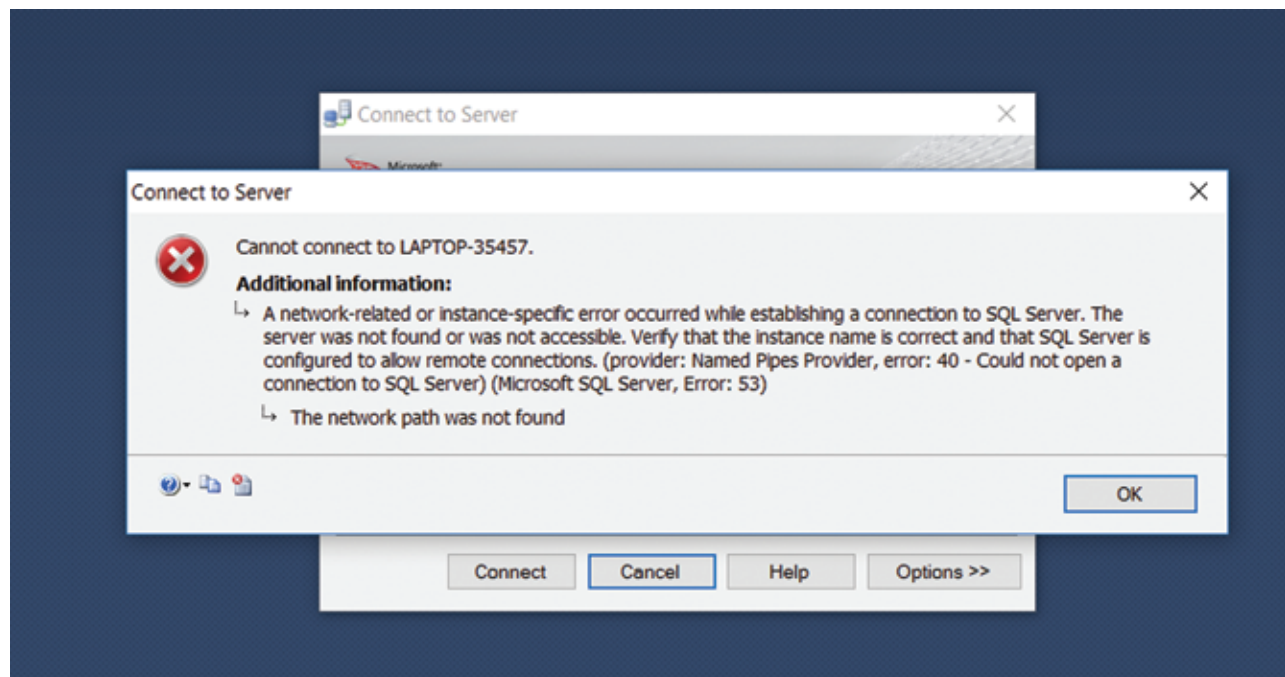


Figure 5

If you experience this error, try selecting “<Browse for more...>” as seen in Figure 6 below.

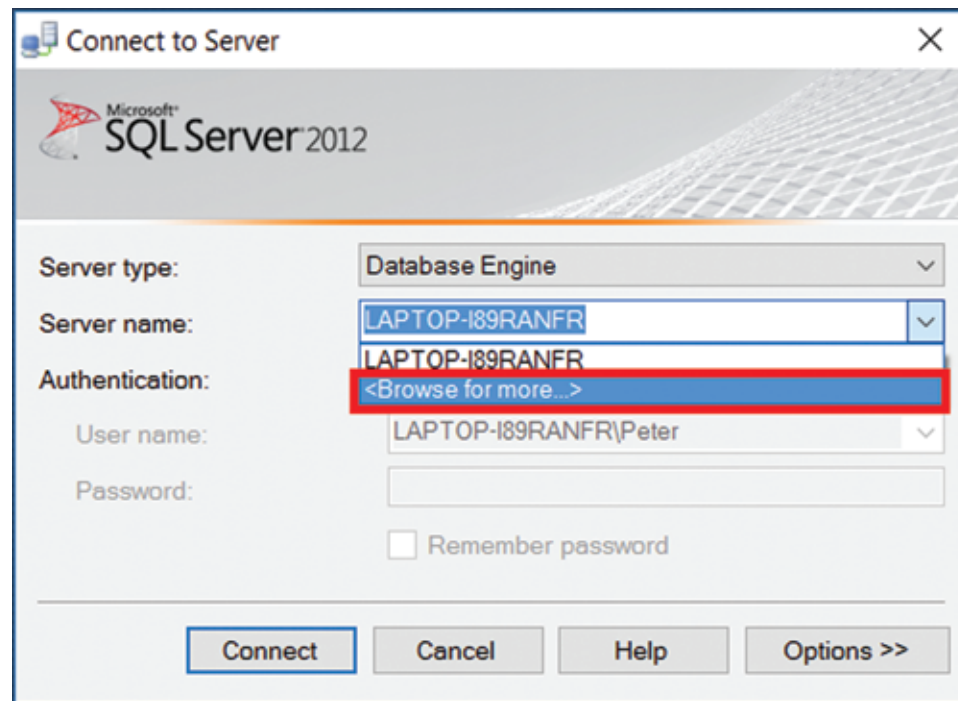


Figure 6

From here navigate to “Database Engine” and select the computer name in the dropdown and then click “OK”. After following these steps try to reconnect using the correct database engine.

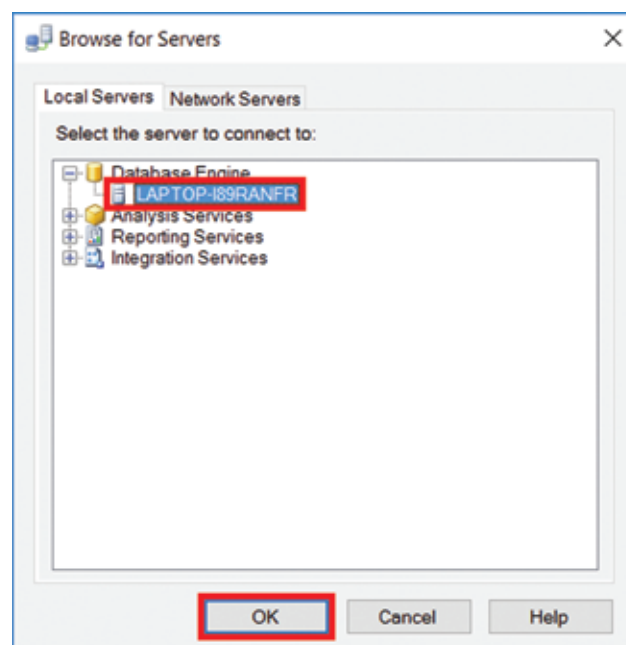
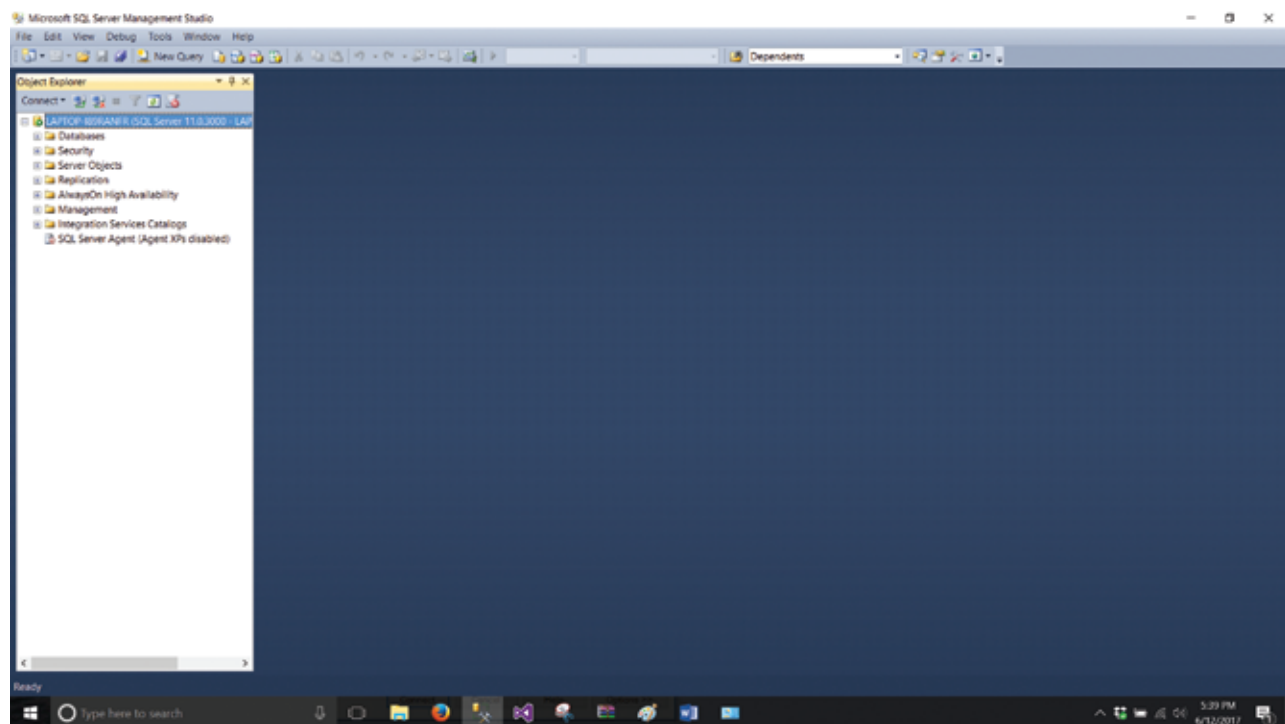


Figure 7

After you successfully connect to SQL Server you should see a screen like that shown below in Figure 8.

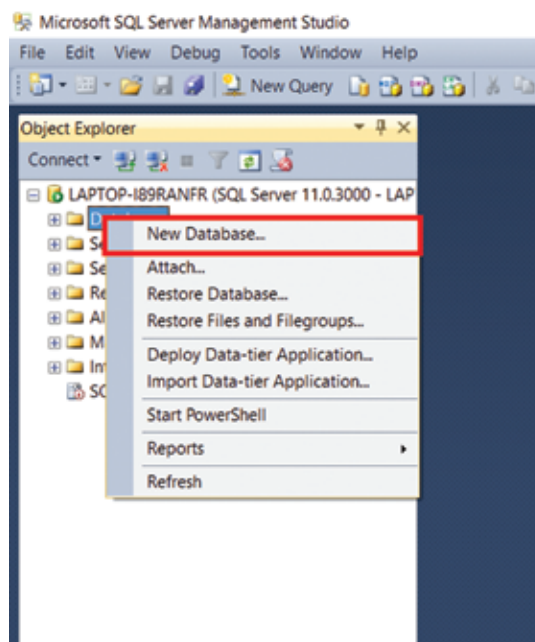


**Figure 8**

## Creating a Database

To become familiar with SQL Server we will create a database using its graphical interface tools. This allows us to create a database without having to input any SQL code.

Let's create a database named "Company". Right click on Databases and select "New Database" as shown in Figure 9.



**Figure 9**



A new pop-up screen will appear as shown in Figure 10. Enter “Company” as the database name. As you can see highlighted, there are two database file names. One is the name of the database and the other is the name of the log file that will contain information on transactions that apply to the database so that database administrators can monitor the database. You need not worry about that as it will do it automatically.

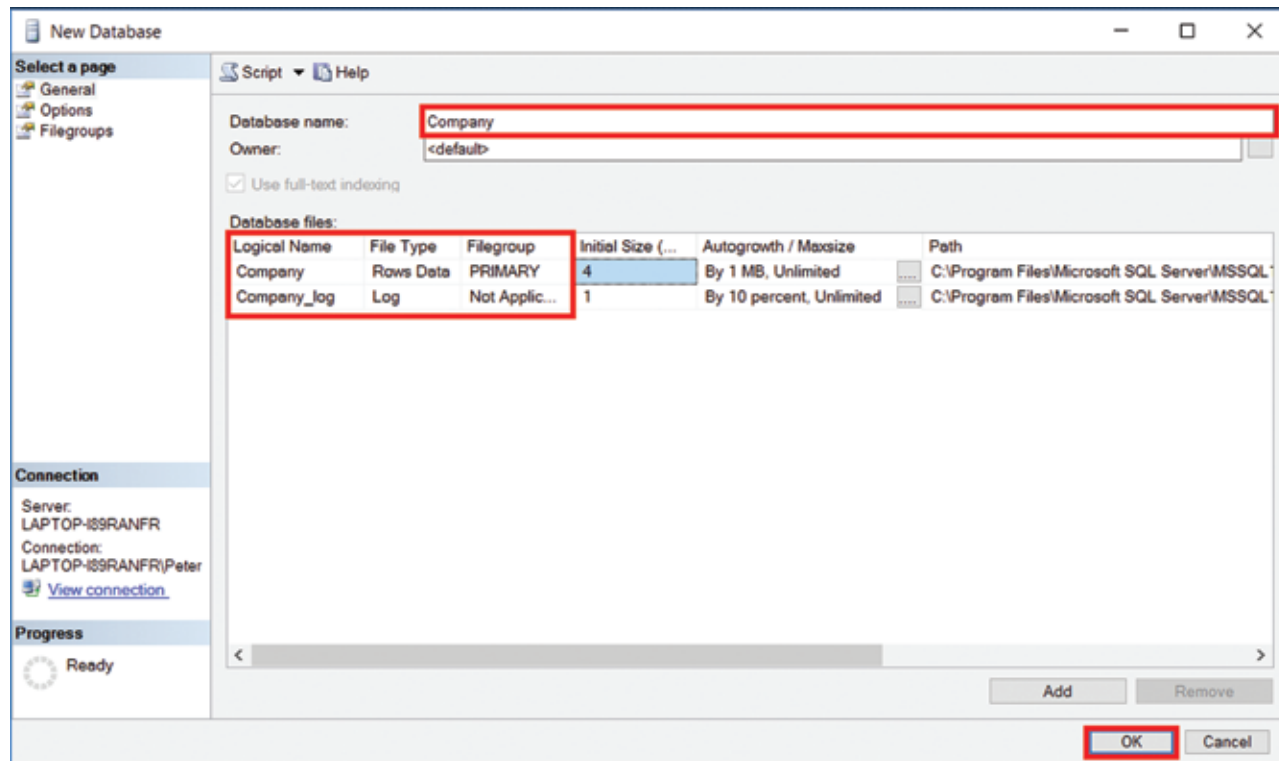


Figure 10

**VERY IMPORTANT: DO NOT PRESS “ADD.”** This will prompt you to add another database file and that’s not what we want to do. All we want to do is type in the database name and click the “OK” button.

Chances are, you will **not** see the Company database that was created in the object browser. Therefore, right click on “Databases” and select “Refresh” as shown in Figure 11. Once you do you should see the company database as shown in Figure 12. You could expand it to see all the object folders also shown in Figure 12.

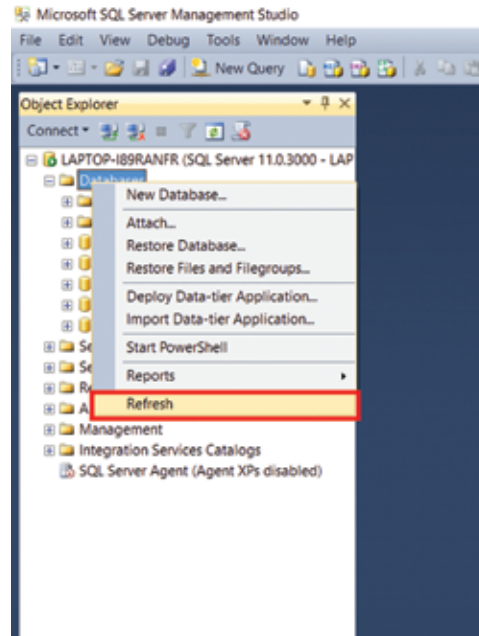


Figure 11

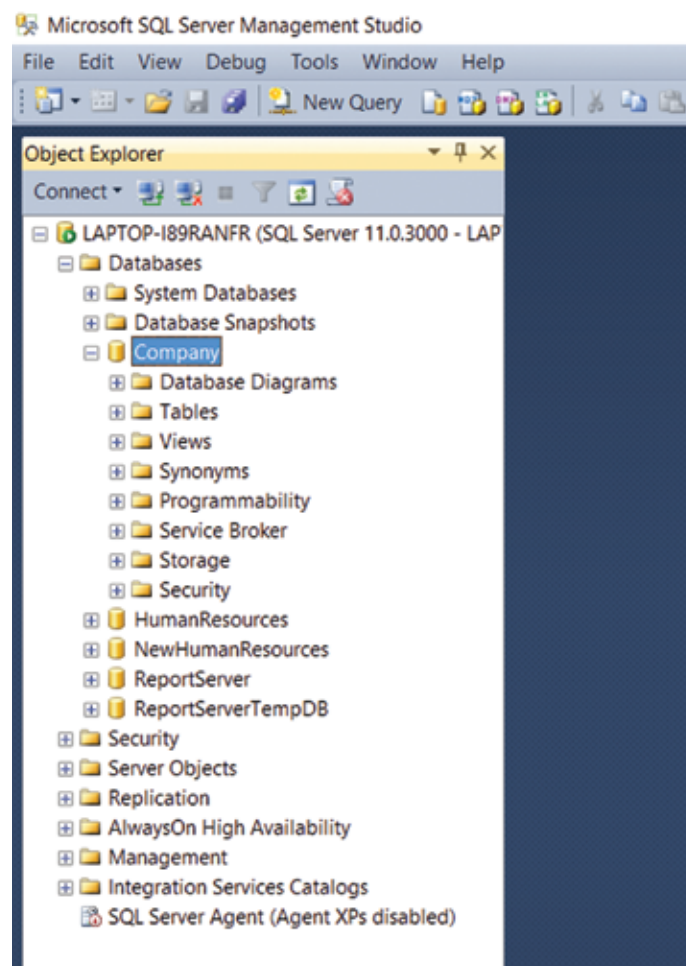


Figure 12



We could have created the Company database by using code as well with the following statement.

```
CREATE DATABASE COMPANY
```

► **NOTE:** SQL is not case sensitive. I use uppercase to make it stand out from other writing as well as clarity.

If you want to try using code to create a database select “New Query” as shown in the red rectangle in Figure 13.

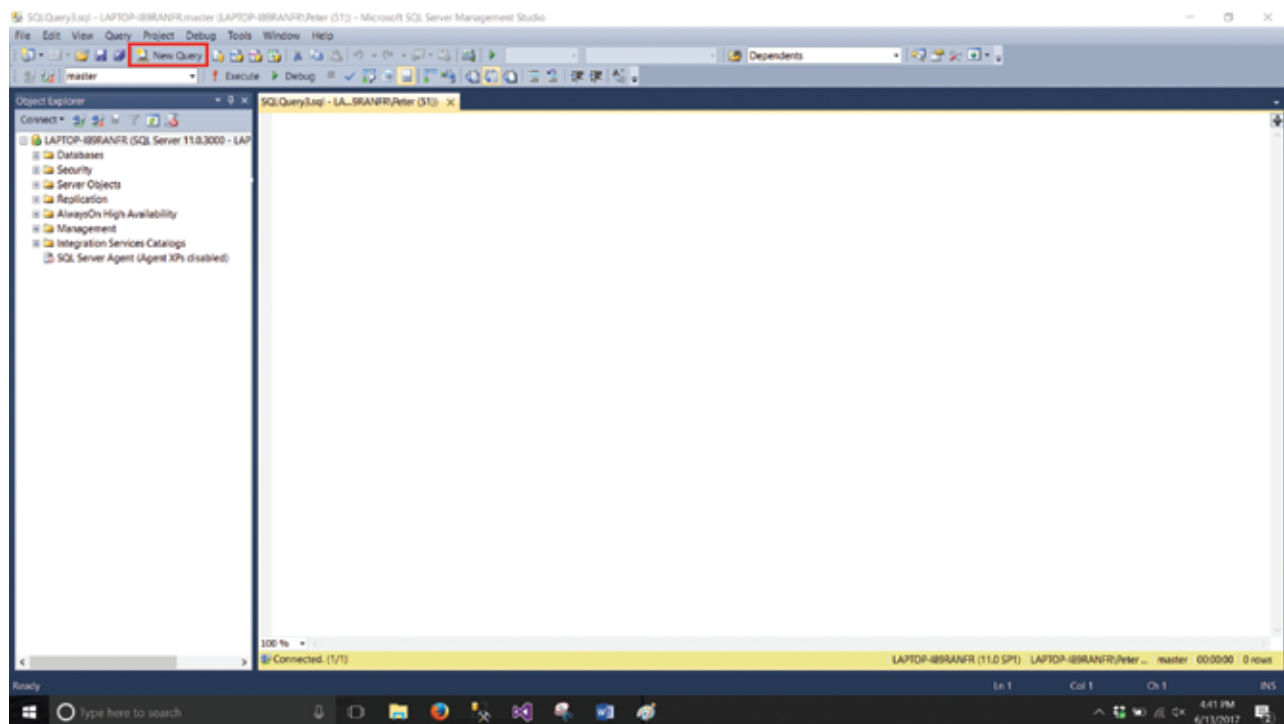


Figure 13

After selecting “New Query” a blank area will appear where code can be written. Since we already created a Company database I used code to create a Company2 database. See Figure 14.

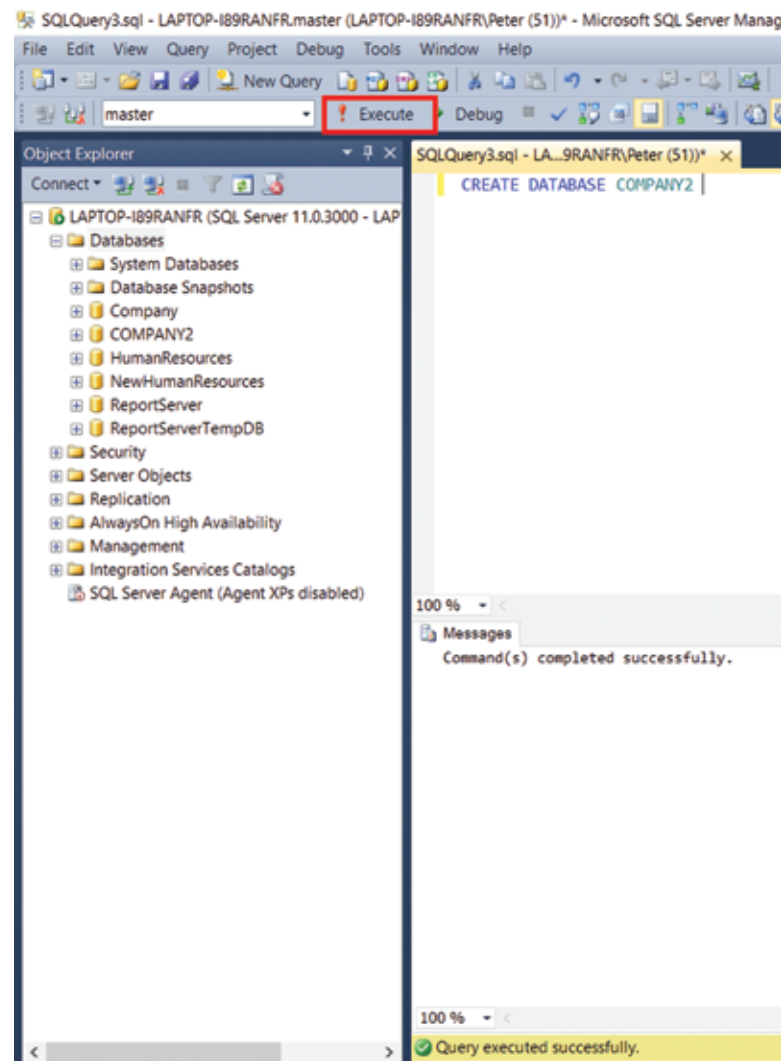


Figure 14

Select the “CREATE DATABASE COMPANY2” and click the “Execute button” shown in the red rectangle in Figure 14.

Now if you do a refresh as instructed earlier in this lesson your Company2 database will show up in the object panel as shown in Figure 15.

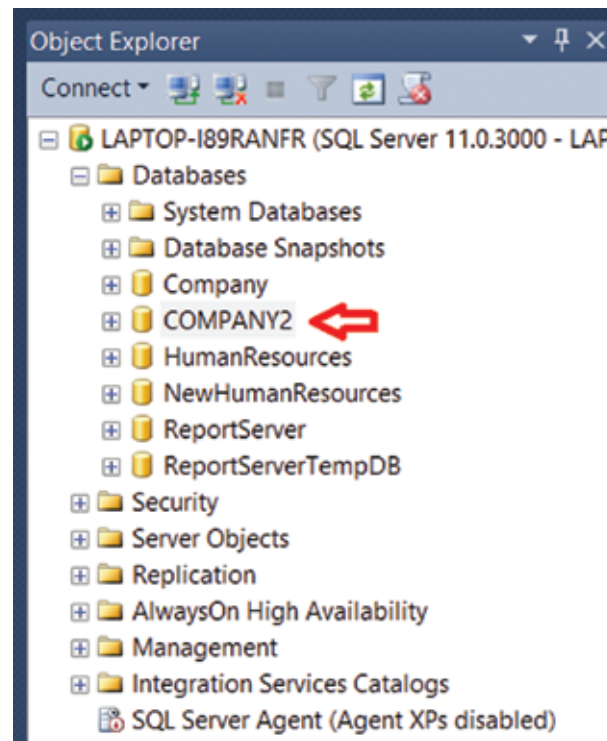


Figure 15

To delete the Company2 database we can simply issue the following command.

```
DROP DATABASE COMPANY2
```

- **NOTE:** When you have two different lines of code on the same query page as seen in Figure 17, you must highlight the code you want to execute with your mouse as I did. Simply clicking “execute” without highlighting a specific line of code, will execute all the code on the page at once. Being that the two different lines of code on Figure 16 contradict each other, this will result in an error message.

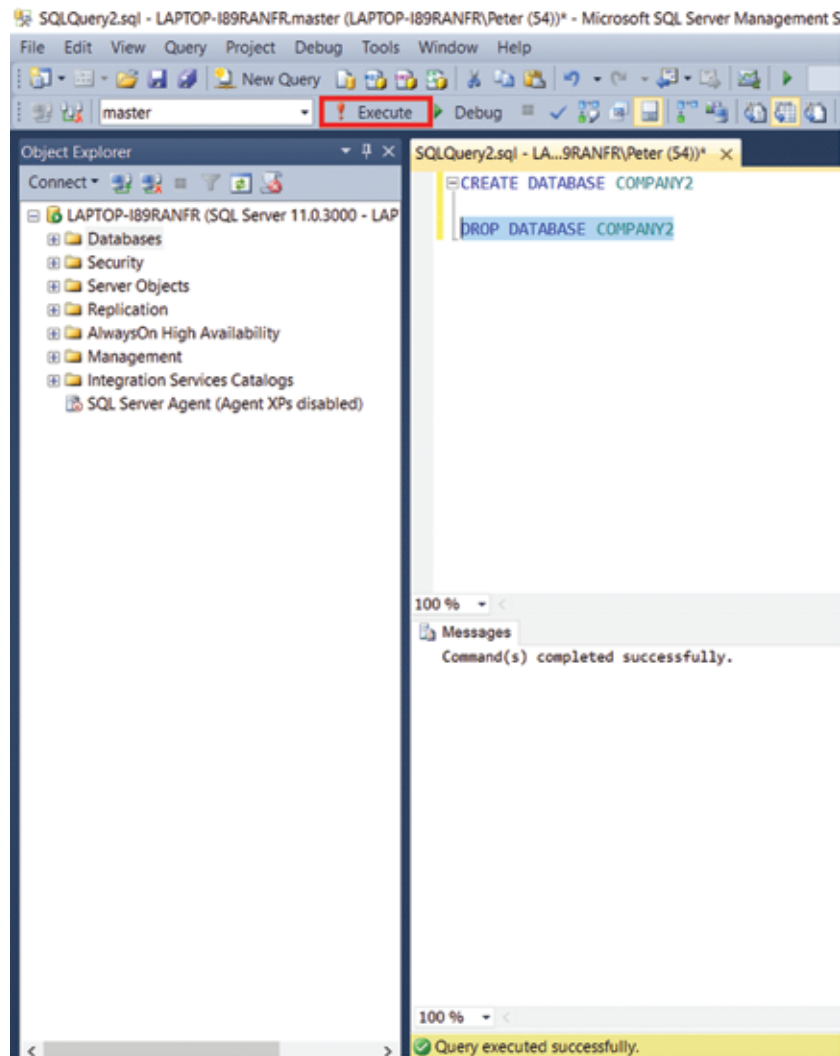


Figure 16

Refresh the database and your screen should look like that in Figure 17 below where Company2 no longer exists.

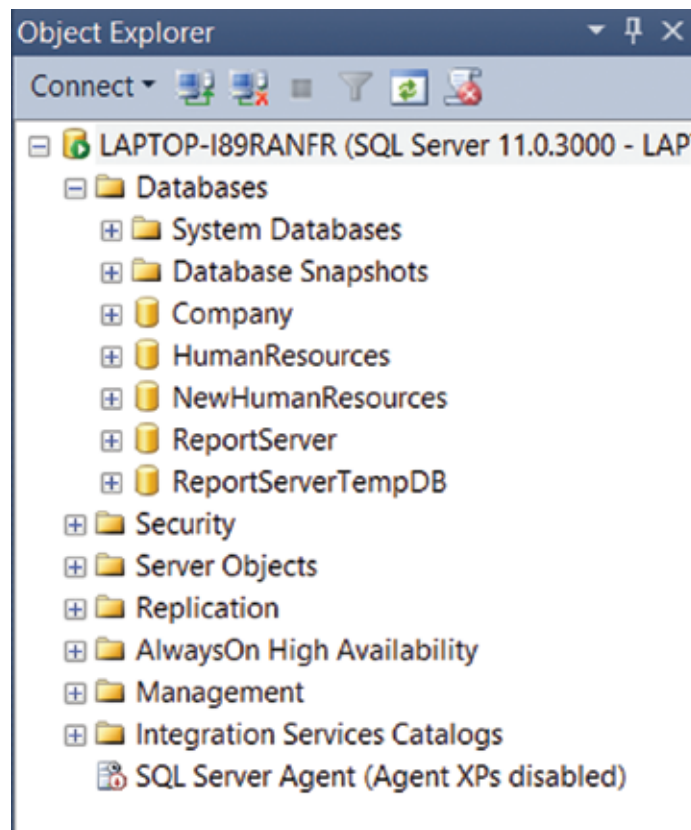


Figure 17

## Database Objects

If you expand the Company database, you will see many folders as shown in Figure 18.

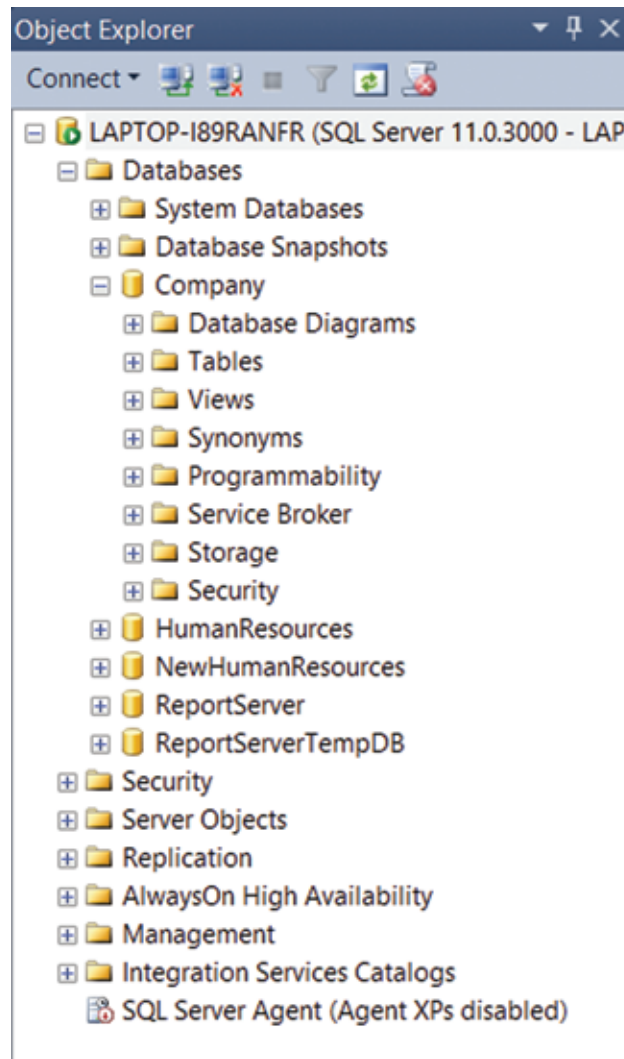


Figure 18

These folders are where database objects are stored. We will focus on two database objects in this chapter, tables and database diagrams.

Let's create a table called Employees in the Company database. But before we continue let's look at the type of data types used for storing data.



### Character Data Types

| Type             | SQL Server Designation | Description  |
|------------------|------------------------|--|
| Fixed            | CHAR(size)             | Fixed-length character data of <i>size</i> characters padded with spaces. Maximum <i>size</i> is 8000 characters.  |
| Fixed Unicode    | NCHAR(size)            | Fixed-length Unicode character data of <i>size</i> characters padded with spaces. Maximum <i>size</i> is 4000 characters. Unicode character set accepts international characters. Unless you are expecting characters from other countries it should be avoided as it uses more storage than the standard character set.   |
| Variable         | VARCHAR(size)          | Variable-length character data of <i>size</i> characters. Maximum <i>size</i> is 8000 characters. VARCHAR is a resourceful data type to use when the size of a column is not distinctly known because it only uses space that is necessary. For example, let's say we provide 25 characters for last name and one of the names we are saving is Smith. Well the VARCHAR character set takes up 5 characters for the name and one character to determine the size for a total of 6. Whereas the CHAR data type would utilize all 25 characters. |
| Variable Unicode | NVARCHAR(size)         | Variable-length Unicode character data of <i>size</i> characters. Maximum <i>size</i> is 4000 characters.  |

### Binary Data Types

| SQL Server Designation | Description   |
|------------------------|---|
| BINARY                 | Holds fixed-length binary data with a length from 1 to 8000 bytes.  |
| VARBINARY              | Stores variable-length binary data with a length of 1 to 8000 bytes. <i>VARBINARY(max)</i> stores up to $2^{31}-1$ bytes maximum. |
| IMAGE                  | Variable length binary data up to 2 billion bytes in length.  |

### Numeric Data Types

| Data Type          | Value Range  | Description  |
|--------------------|--|--|
| TINYINT            | 0 to 255   | Very small, positive integers.   |
| SMALLINT           | −32,768 to +32,767                                     | Small positive or negative integers.   |
| INT                | −2 billion to +2 billion (pprox..)                     | Typical large positive or negative integers.   |
| BIGINT             | −2 <sup>63</sup> to +2 <sup>63</sup>                   | Monster numbers that are not often used for anything practical.                                |
| DECIMAL or NUMERIC | −10 <sup>38</sup> to +10 <sup>38</sup>                 | Precise numbers consisting of fractional portions—56.38 for example.                           |
| SMALLMONEY         | −214,748.3648 to +214,748.3647                         | Storing smaller currency values accurate to a ten-thousandth of the monetary unit represented. |
| MONEY              | −922,337,203,685,477.5808 to +922,337,203,685,477.5807 | Storing larger currency values accurate to a ten-thousandth of the monetary unit represented.  |
| BIT                | 0 or 1   | Represents off or on, false or true. Used for any two-valued field.                            |
| FLOAT/REAL         | −1.79 × 10 <sup>±38</sup> to +2.23 × 10 <sup>±38</sup> | Represents very small or very large numbers used in scientific calculations.                   |

### Tables

Let's create a table called Employees in the Company database. We are going to create the Employees table two ways. First, we will use code and then we will create it using the graphical user interface. Before we start we need to make sure we're in the proper database. Select the Company database from the drop-down selection box as shown in the red rectangle in Figure 19.

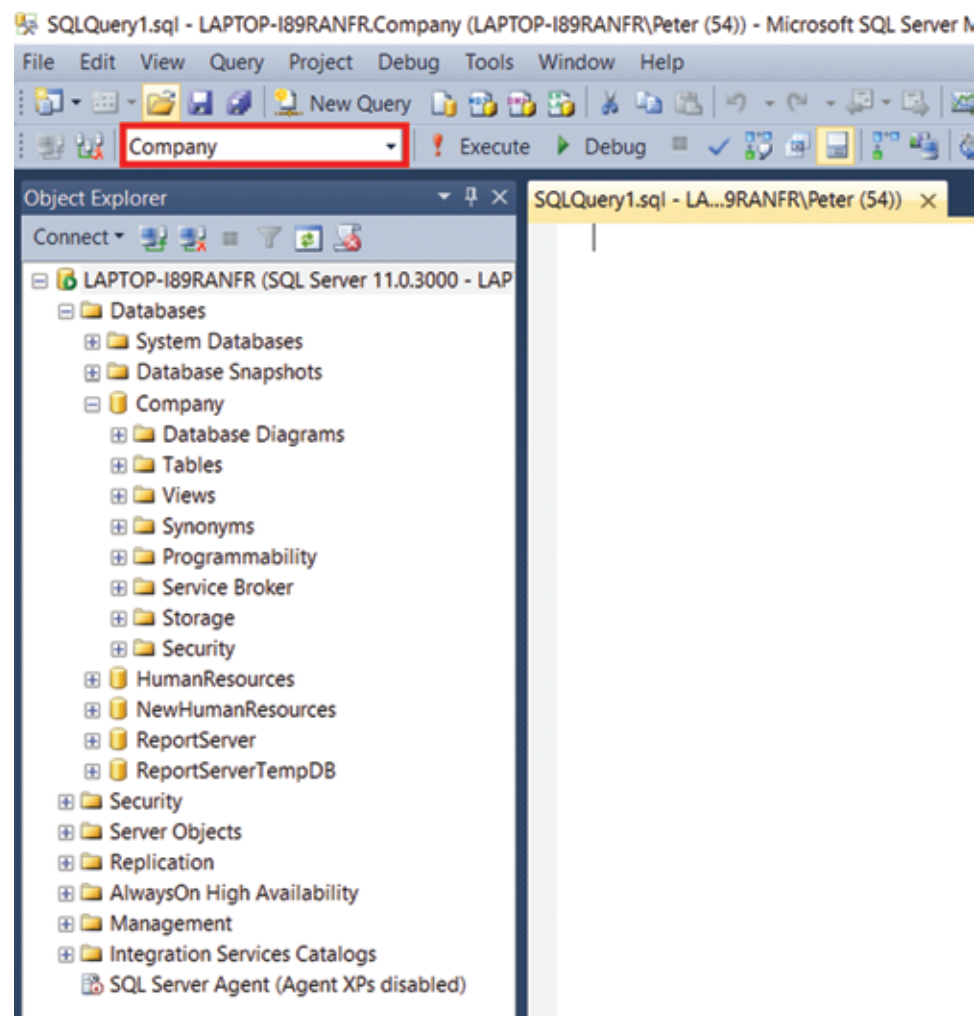


Figure 19

- **NOTE:** Not being in the correct database is often the source of a lot of errors when trying to execute code. If you try to execute code in the wrong database you will receive an error like the one below.

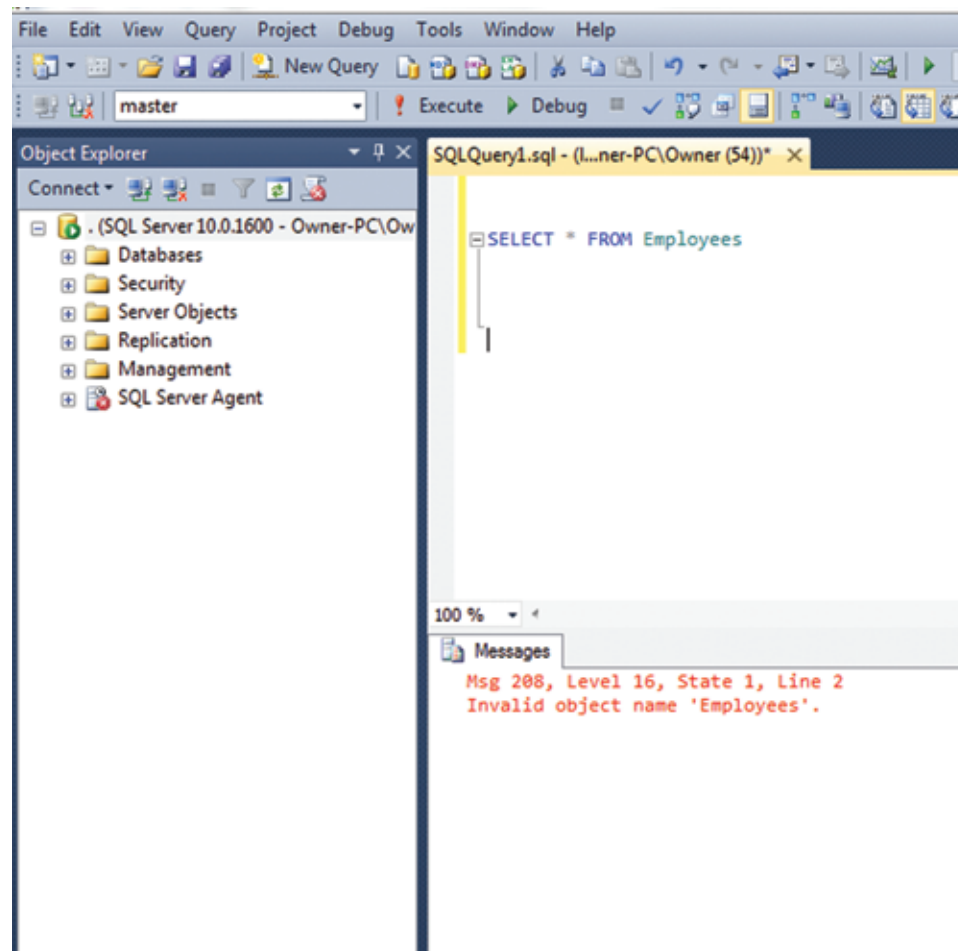


Figure 20

SQL defaults to the “master” database. So always make sure that you change it to the database that you’re working with. When you receive an error, don’t panic. Just check to see if you’re in the correct database and check the spelling of your code.

Below is the code to create the table and the results are shown in Figure 21. An explanation of the code follows.

USE Company

```
CREATE TABLE Employees(
    EmployeeID int IDENTITY(1,1) NOT NULL,
    FirstName varchar(50) NOT NULL,
    LastName varchar(50) NOT NULL,
    DOB date NULL,
    Title varchar (25) NULL,
    CONSTRAINT PK_Employees PRIMARY KEY (EmployeeID ASC))
```

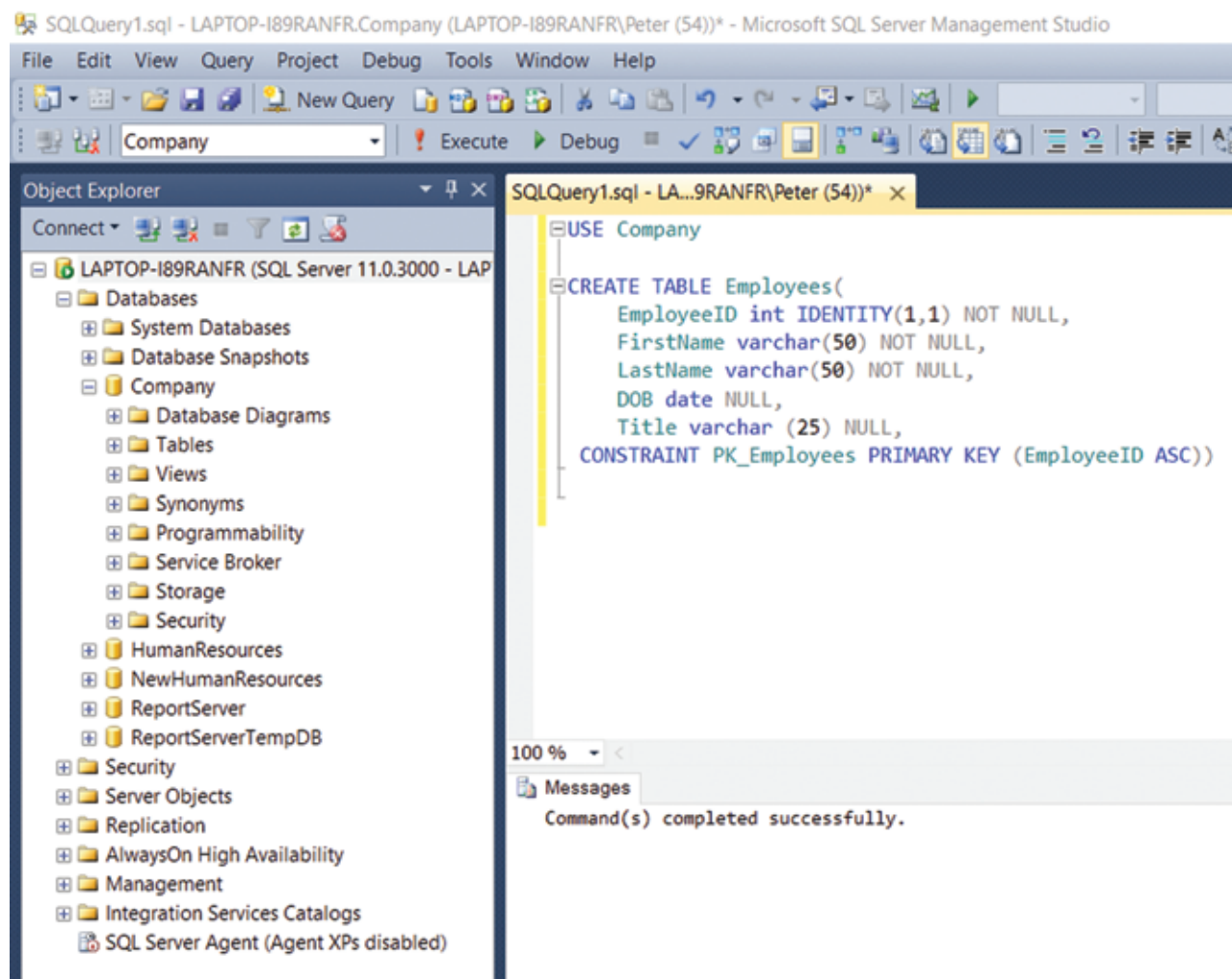


Figure 21

The first line of the code “USE Company” indicates that we want to create this in the Company database. You can also avoid using this statement by selecting the Company database from the drop-down selection box as discussed earlier. The table below explains the rest of the statements.

| Statements  | Description   |
|---|---|
| <code>CREATE TABLE Employees(</code>  | Communicates to SQL Server that a table is going to be created. Notice the opening parenthesis which is required to let SQL Server know that everything in the parenthesis is going to be part of the table.  |
| <code>EmployeeID int IDENTITY(1,1)<br/>NOT NULL,</code>                               | EmployeeID is going to be the unique identifier and it will be an integer. IDENTITY is used so that the primary key will be automatically generated starting with the number 1 and incrementing by one. There must always be a value in EmployeeID so it is not null. |
| <code>FirstName varchar(50) NOT<br/>NULL,</code>                                      | First name of employee and it must contain a value.   |
| <code>LastName varchar(50) NOT<br/>NULL,</code>                                       | Last name of employee and it must contain a value.  |
| <code>DOB date NULL,</code>   | Date of birth is a date datatype and it is not required so it can be null.  |
| <code>Title varchar (25) NULL,</code>   | Position title of the employee and it is not required so it can be null.  |
| <code>CONSTRAINT PK_Employees<br/>PRIMARY KEY (EmployeeID<br/>ASC)) ON PRIMARY</code> | This line communicates to SQL Server that the EmployeeID is a primary key and will be stored in ascending order. The name of the primary key is PK_Employees. This name is used for SQL Server to help manage the database.   |



After running the create table statement refresh the COMPANY database and drill down to see the Employees table and then drill down again to see the fields (columns) you created as shown in Figure 22.

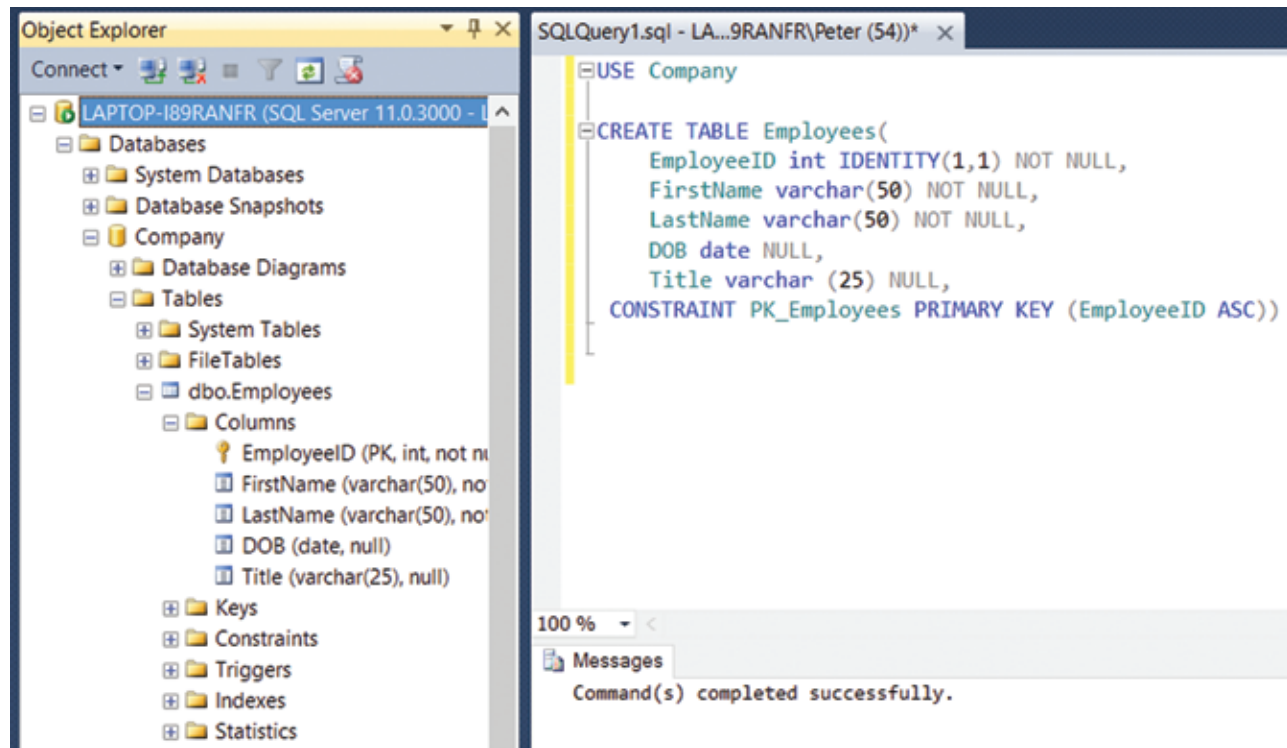


Figure 22

Ok, let's now create the Employees table using the graphical user interface. First let's drop (delete) the table because we can't create a table with the same name. See Figure 23, to see the code for dropping the table. Once you run the code, do a refresh and you'll see the table is no longer there.

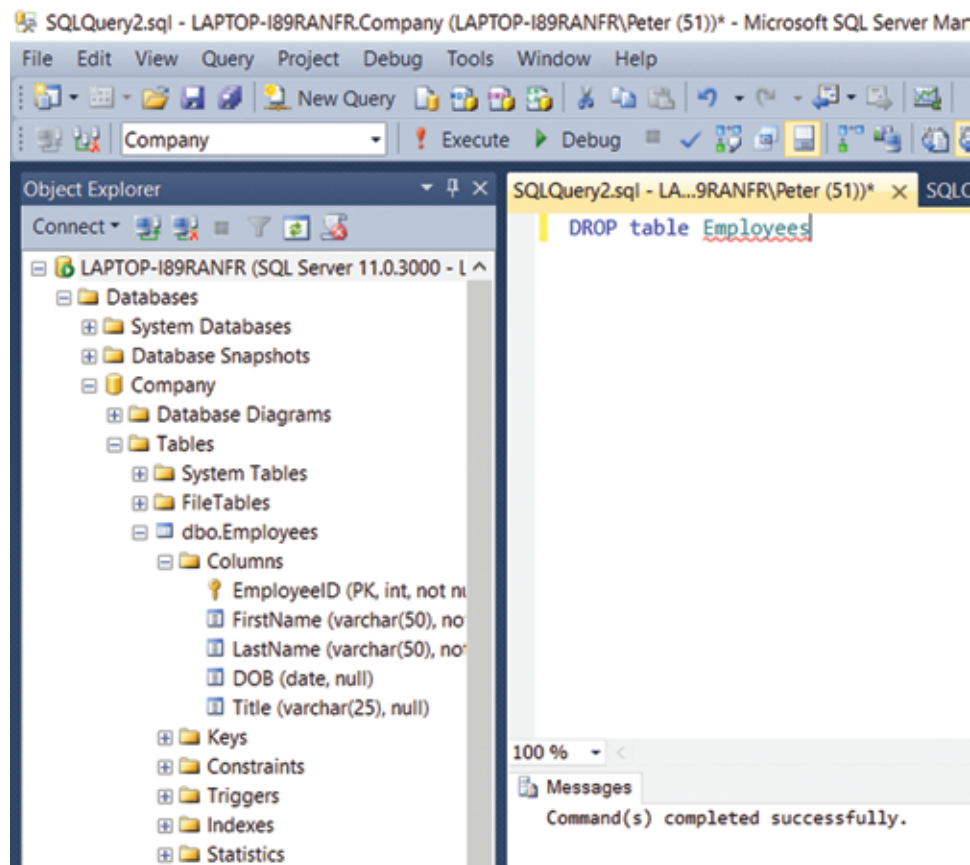


Figure 23

Now that we have dropped the Employees table, we're free to recreate the table using the graphical user interface.

To recreate the Employees table using the graphical user interface, right click on "Tables" in the object explorer and select "New Table" as shown in Figure 24.

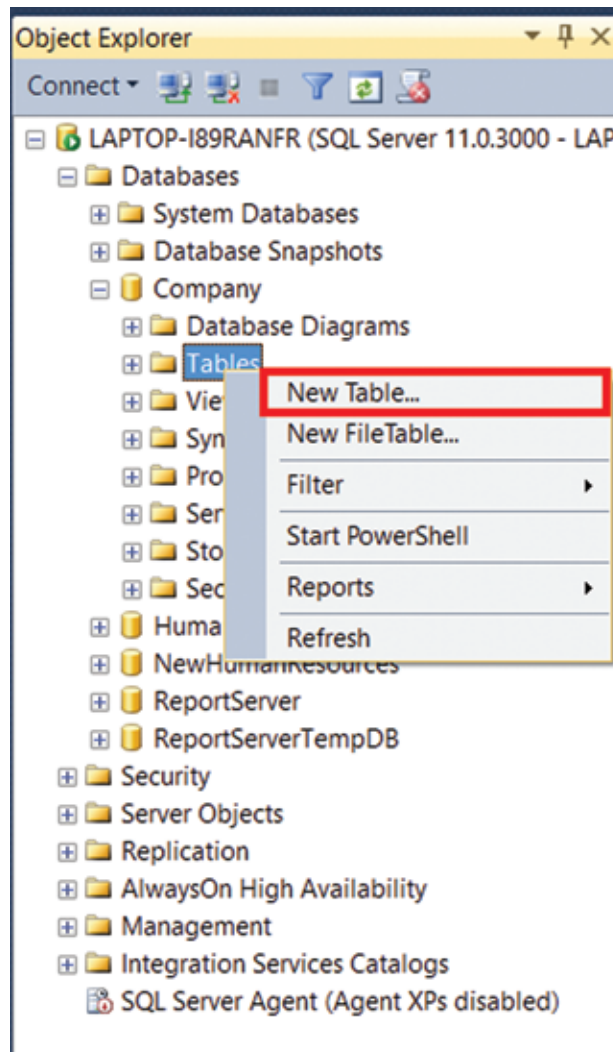


Figure 24

The table design screen will appear as shown in Figure 25. In the table design screen type “EmployeeID”, choose “int” for integer from the drop-down data type box and then make it a primary key by selecting the key on the tool bar. Once you make the field a primary key the “Allow Nulls” will be unchecked automatically because primary keys can never allow nulls.

Next move down to Column Properties and expand Identity Specification to set the column as an Identity column starting with the number one and incrementing by one. See Figure 25. This will allow the table to automatically generate a unique number sequentially every time a row of data is added.

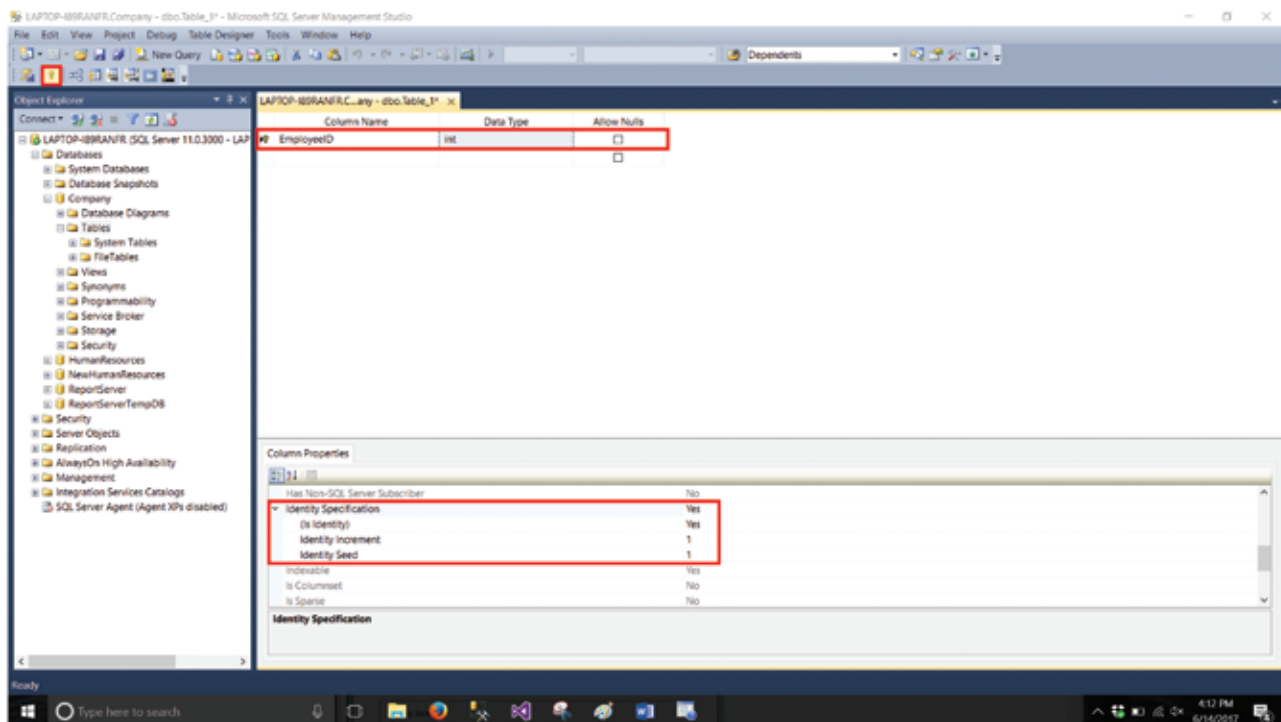


Figure 25

Add the rest of the fields as you see in Figure 26. Notice that I unchecked the “Allow Nulls” check boxes for first and last names but left the other fields checked. That’s because I’m assuming as a business rule that first and last names are required at the very least. Date of birth and title, on the other hand, isn’t as crucial and an employee can still be inserted into the database without that information.

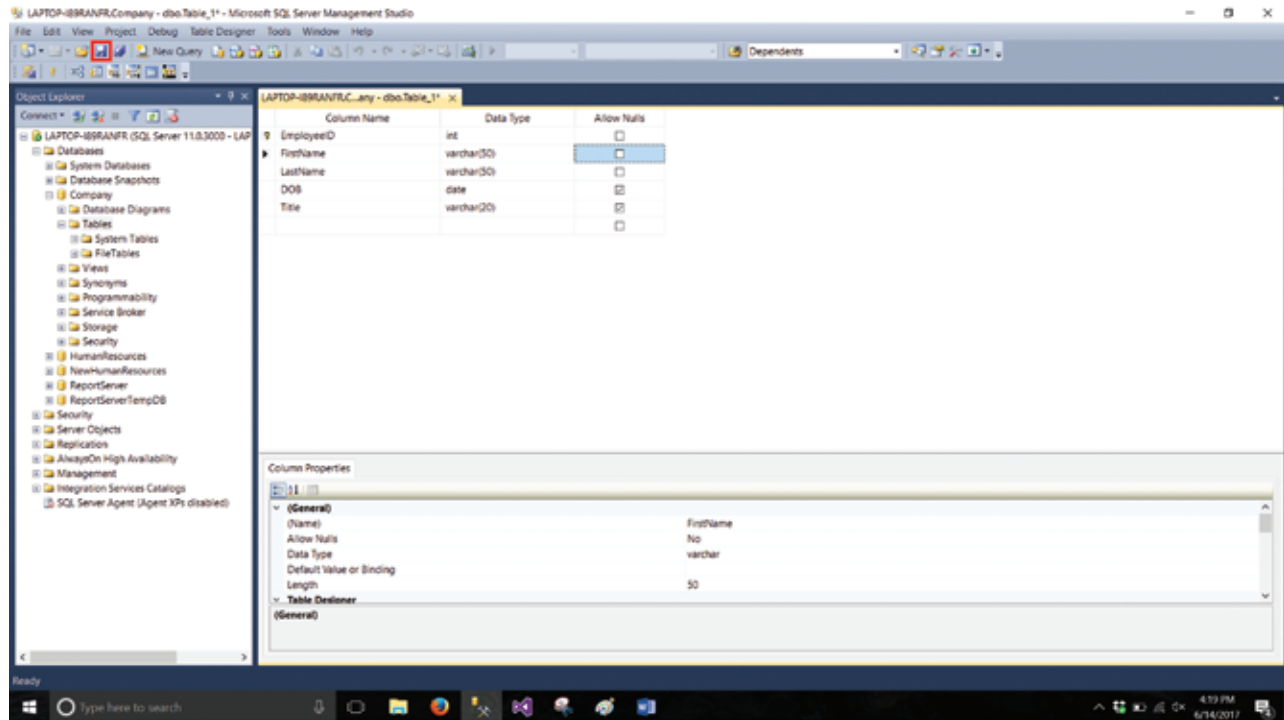


Figure 26

Click the “Save” button and name the table “Employees” in the pop-up box as shown in Figure 27.

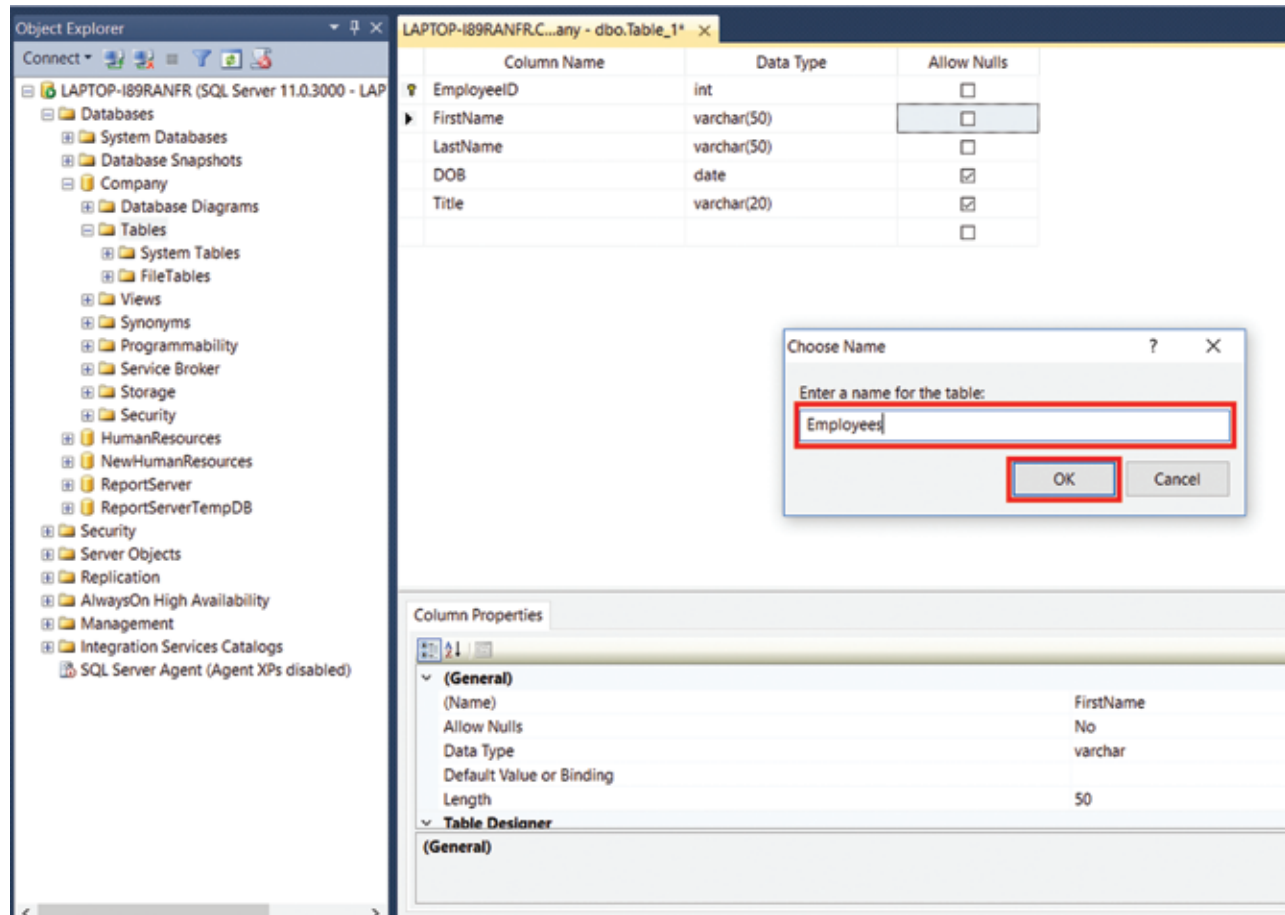


Figure 27



After saving and naming your database, you can close the database design screen, refresh, and see your new table as shown in Figure 28.

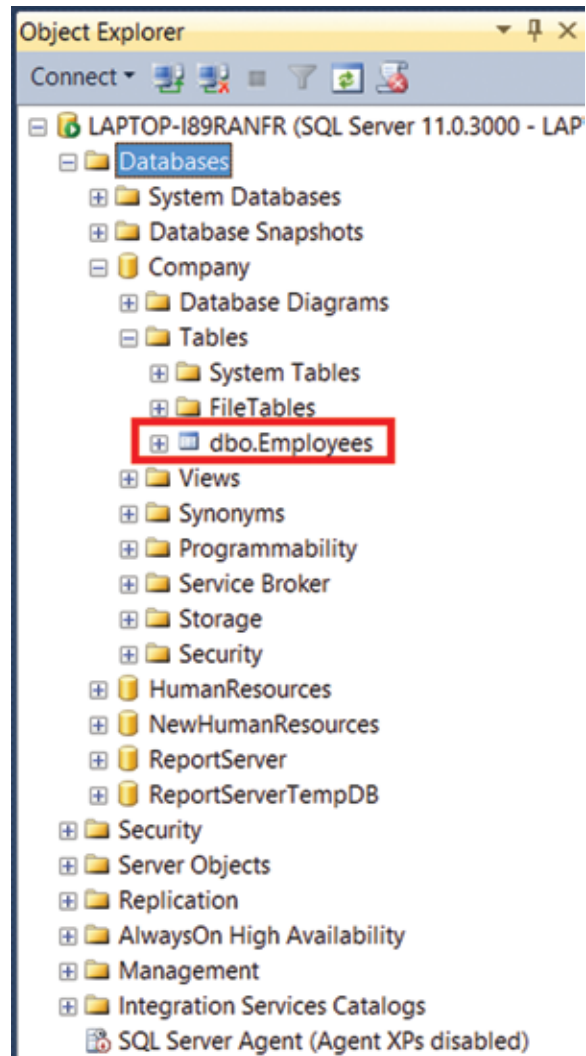


Figure 28

## Database Diagram

In this section we will create a database diagram also called an entity-relationship diagram (ERD). We discussed entity-relationships in Chapter 1. We will first create the physical database and tables. Then we will use the database diagram tool to build physical relationships.

First, let's create a new database called "COLLEGE." You can either create it using the graphical user interface or simply issue the command shown in Figure 29.

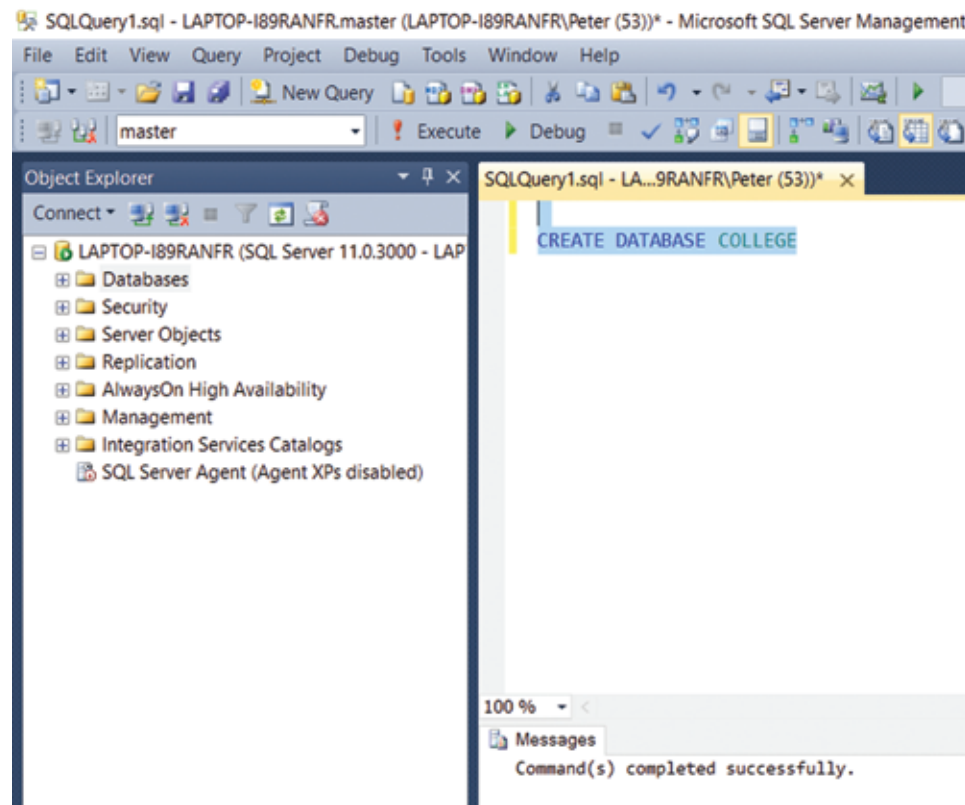


Figure 29

Let's build a Teachers table. Select Tables from the browser, right click, and select New Table. You should see the screen in Figure 30.

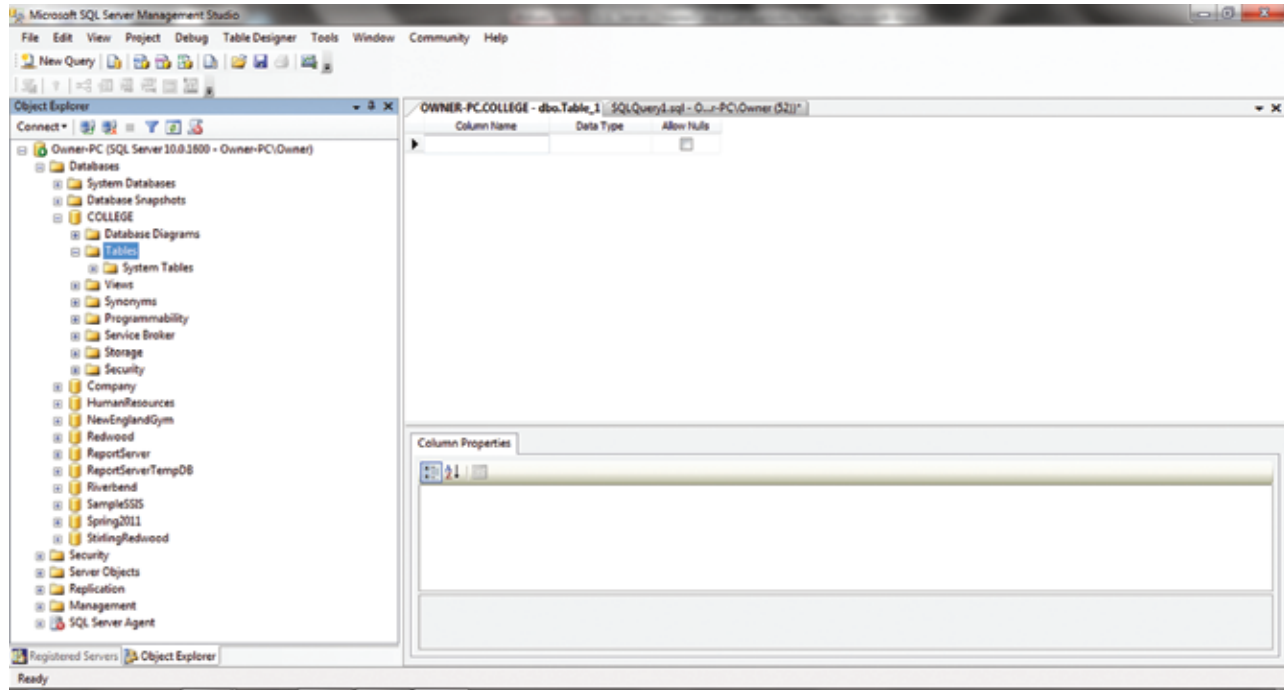


Figure 30

Create the table as shown in Figure 31. Set the primary key by right clicking on TeacherID as selecting Primary Key.

- **NOTE:** You may see “nvarchar” as a default type. Nvarchar is for unicode used to incorporate the national language character set for foreign languages. Since we won’t be doing this stick with varchar. Nvarchar uses up more resources, so don’t use it unnecessarily.

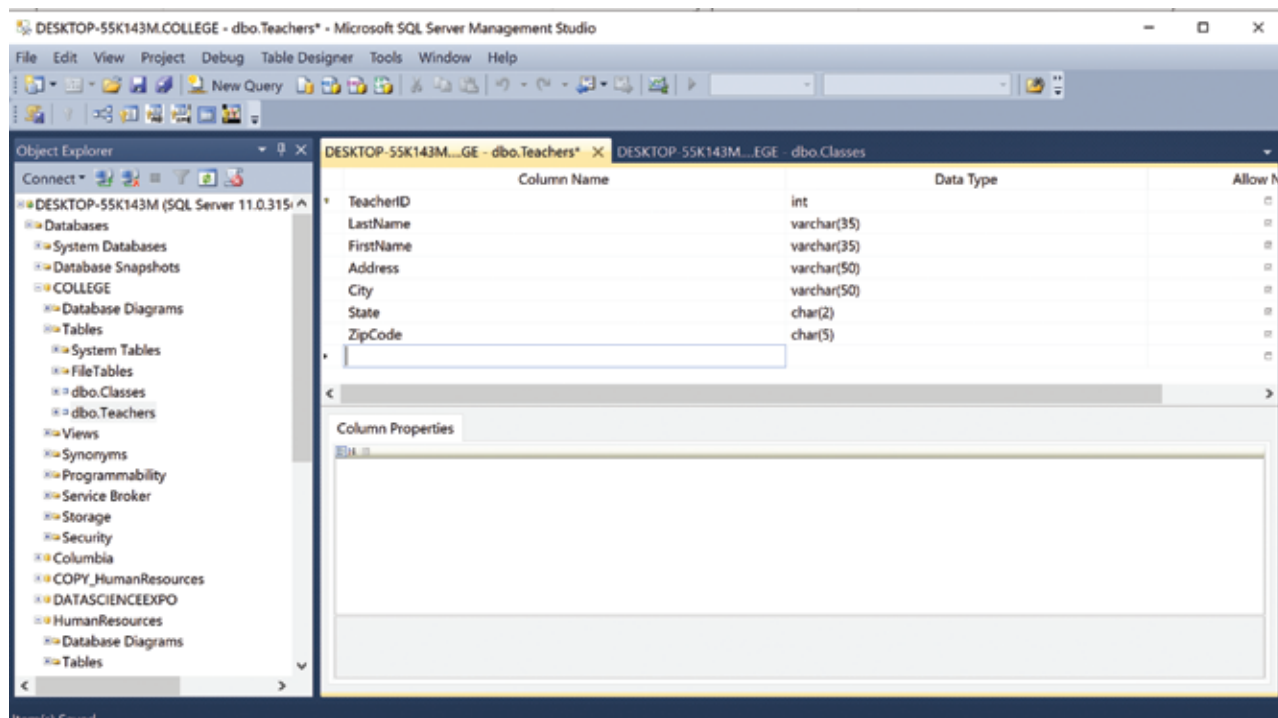


Figure 31

The following columns and data types make up the Teachers table.

| Column Name | Data Type   | Explanation  |
|-------------|-------------|--|
| TeacherID   | Integer     | Numeric primary key that will auto increment by 1.   |
| LastName    | Varchar(35) | Variable data type with a max size of 35 but will not take 35 characters if name is less than 35. This saves on data storage.  |
| FirstName   | Varchar(35) | Same explanation as LastName.  |
| Address     | Varchar(50) | Decided to use 50 characters because address can be lengthier than names.  |
| State       | Char(2)     | Chose Char of a size two because state is always a fixed size 2.   |
| Zip Code    | Char(5)     | Chose Char of a size 5 because state is always a fixed size 5. For any numeric data that cannot be used for calculations, it is always character data.   |
| DOB         | Date        | Although DOB can be stored as character data type all dates should be stored as the date data type. This is very beneficial when performing calculations on dates such as determining ages based on DOB. |

Create another table for Classes as shown below in Figure 32.

## Classes

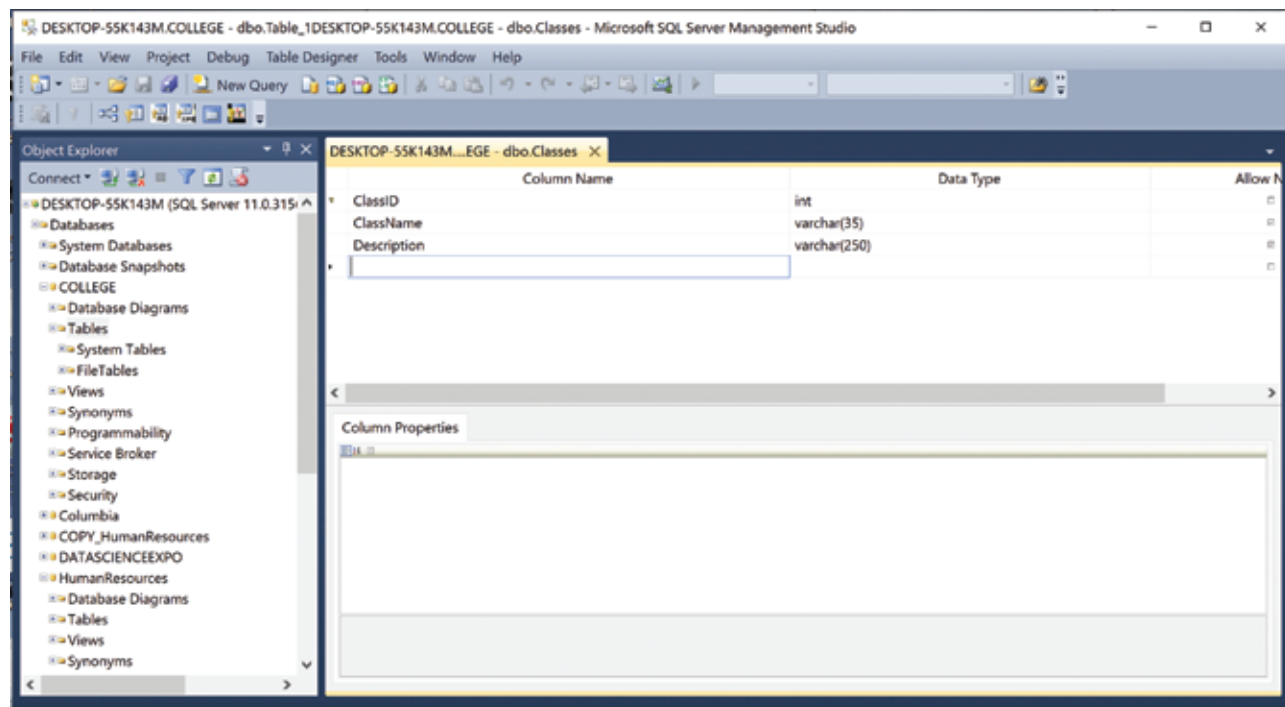


Figure 32

Now since teachers can teach many classes classes can be taught by many teachers, we need a bridge table. Create the bridge table and name it TeachersClasses as shown in Figure 33. This table will be used to show what classes each teacher will teach. Notice that TeachersClasses has two foreign keys. These foreign keys relate back to the Teachers and classes tables. They will be used to form a unique composite primary key. However, since teachers can teach the same class more than once in one or more semesters at different times of the week, we will need semester and day and time combined with the foreign keys to make the composite key unique.

## TeachersClasses

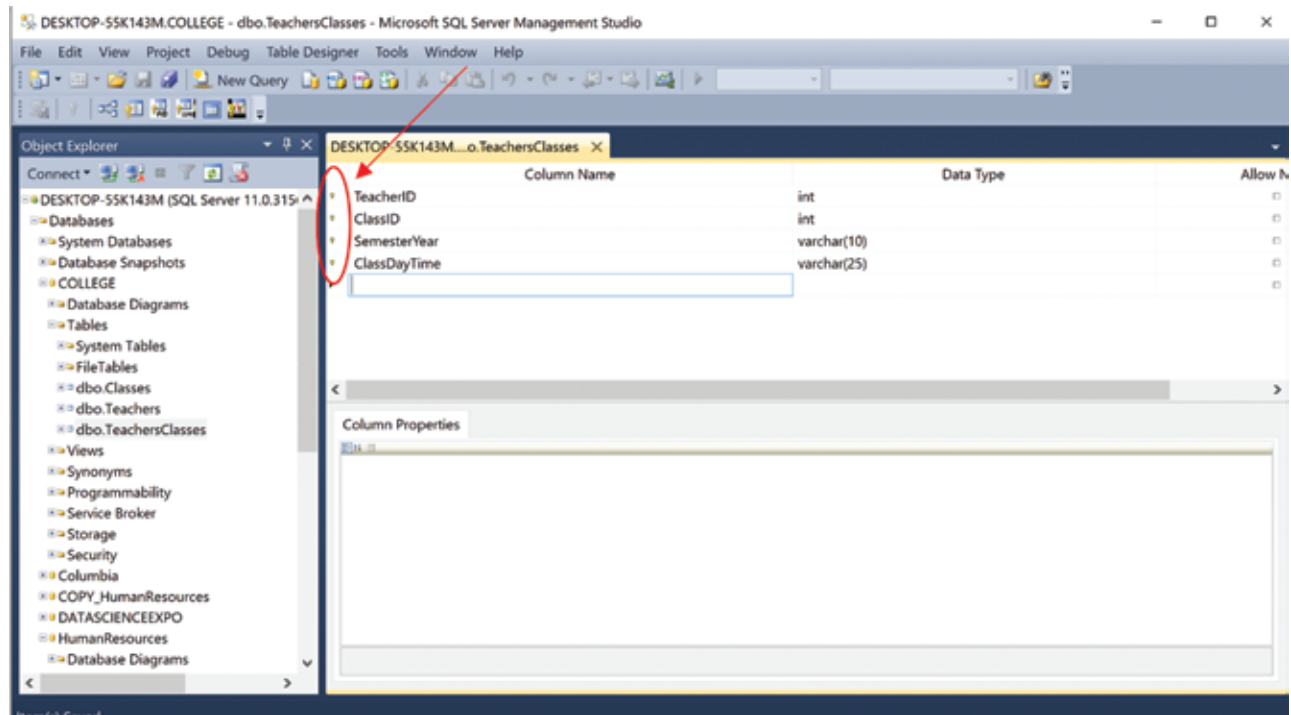


Figure 33

- **NOTE:** To make the composite key press and hold the control key and select TeacherID, ClassID, SemesterYear, and ClassDayTime and then click the primary key button.

Next we will create a diagram showing the tables and their relationships.

## Creating a Database Diagram

Right click on database diagrams under COLLEGE and select new diagram from the menu as shown in Figure 34.

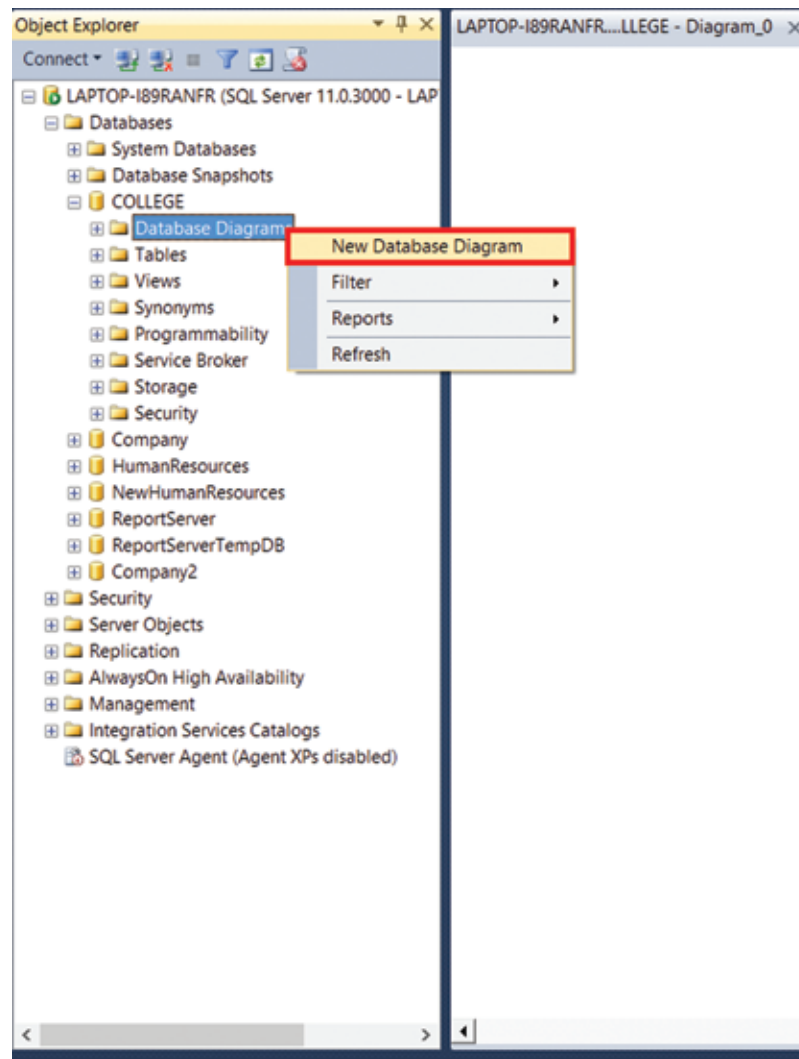


Figure 34

Then highlight and “Add” each table into the database diagram as shown in Figure 35.

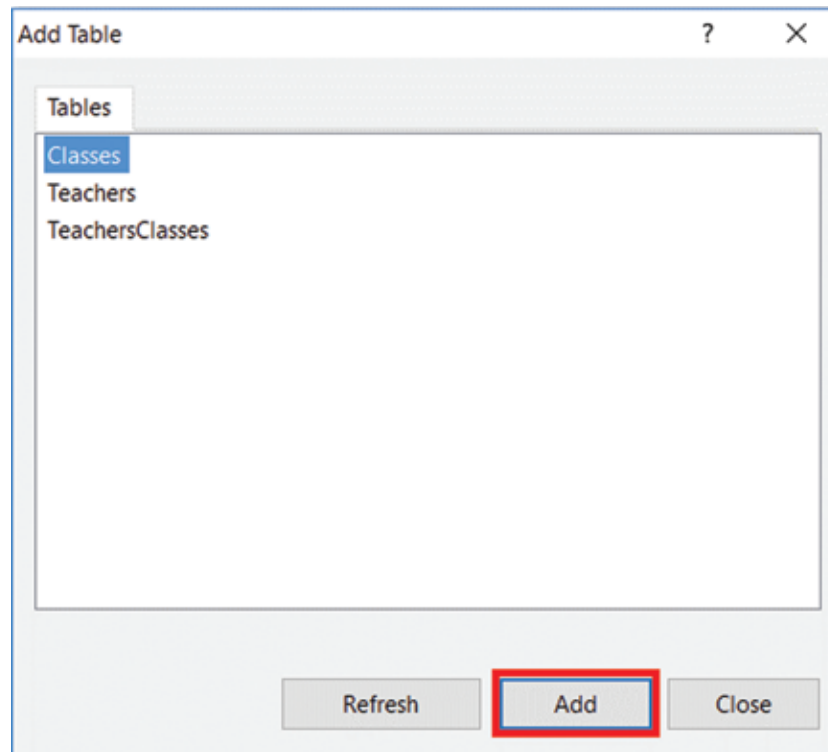


Figure 35

After you’ve added all your tables click “Close” on the “Add Table” window and arrange your tables like that shown in Figure 36 if they’re not that way already.

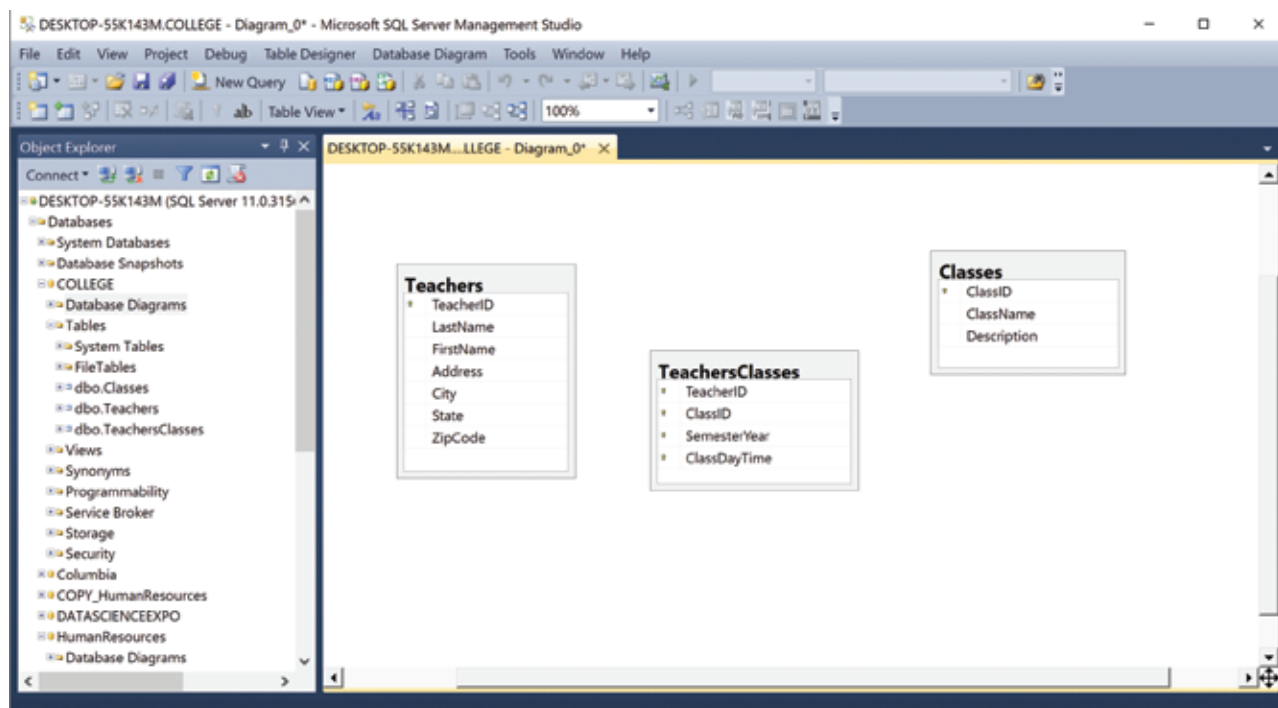


Figure 36



Next, drag the TeacherID from the Teachers table over to the TeacherID of the TeachersClasses table. Once you do you should see the window shown in Figure 37. This window is going to create the one-to-many relationship between the Teachers and TeachersClasses table. SQL Server names the relationship automatically so you don't have to worry about that.

As you can see in Figure 37, The Teachers TeacherID is the primary key while the TeachersClasses TeacherID is the foreign key. Click on "OK" to advance.

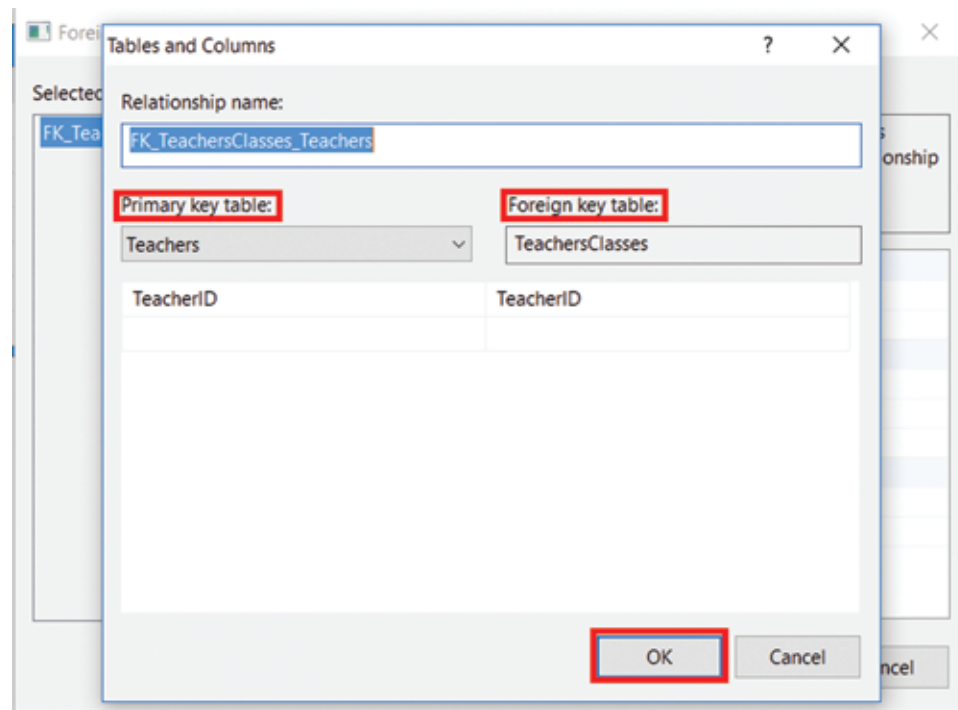


Figure 37

After you click “Ok” on the window shown in Figure 37, a second window will appear as shown in Figure 38. This window is simply confirming the relationship and keeping this window set to its default values will allow the tables to stay updated. Therefore, just click “Ok.”

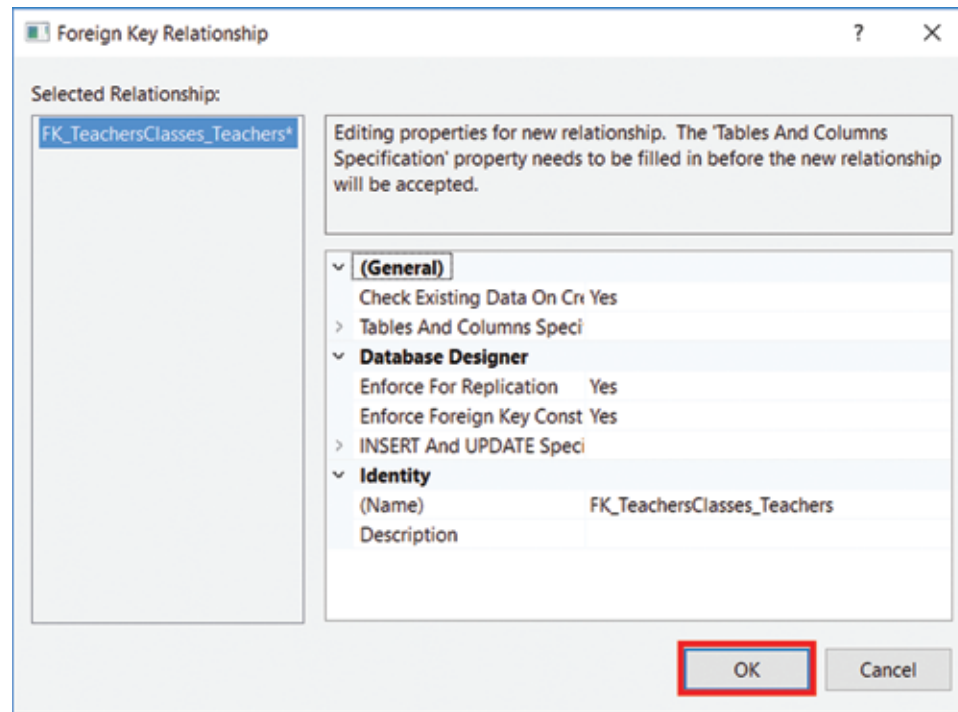


Figure 38

After completing these steps, you should now have a relationship between the Teachers table and the TeachersClasses table as shown in Figure 39. Now, repeat these steps and create a relationship between the Classes table and the TeachersClasses table.

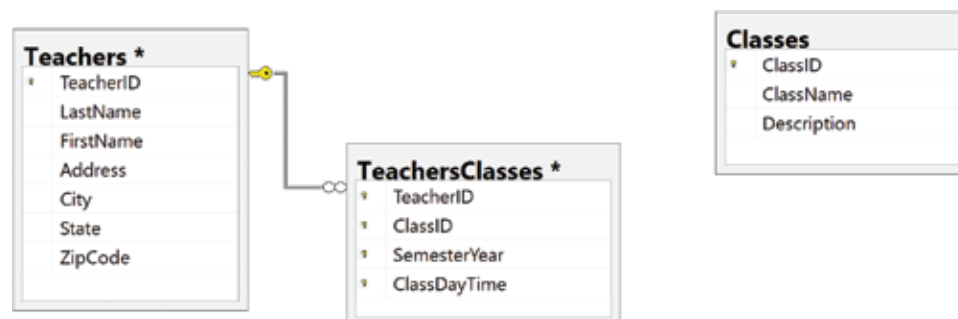


Figure 39

This creates your relationships and creates them so that relational integrity is enforced. For example, TeacherID 100 cannot exist in TeachersClasses unless it exists in the Teachers table. The same goes for Classes. The ClassID in TeachersClasses must exist in Classes. Both Teachers and Classes have a one-to-many relationship with TeachersClasses.

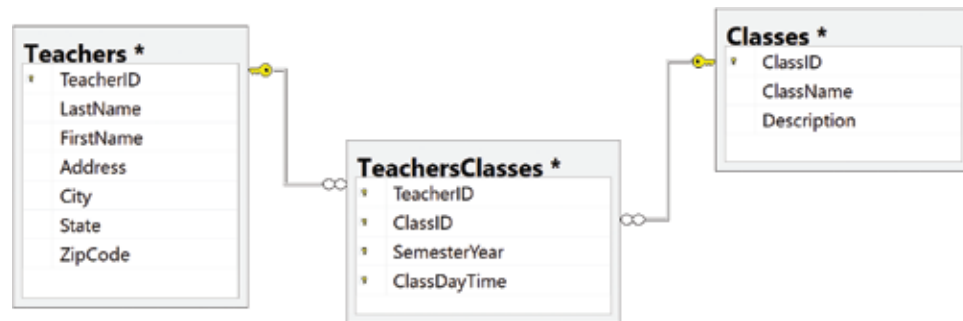


Figure 40

## Summary

This chapter introduced you to the SQL Server and some of its GUI tools. The chapter covered creating databases and tables using SQL code as well as using the GUI tools. Column data types were covered so that you understand the importance of choosing the appropriate data types for various kinds of data. The chapter ended with creating an ERD using SQL Server's diagram tool.

## Exercise

1. Create a SQL database for the normalization exercise you completed in Chapter 1 for doctors and patients.
2. When complete, use the “Snipping Tool” to take a screen shot of the database diagram including tables and relationships.