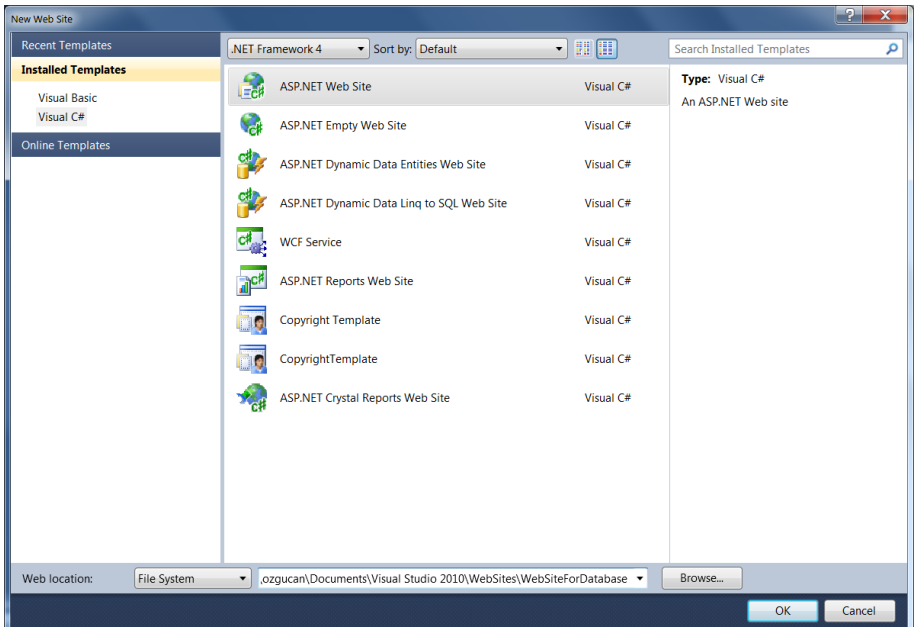
## Connecting to the SQL Server Sample Database

In this exercise you learn how to connect to and work with a database from within VS. To give you something to work with, the sample database contains a few tables. To be able to access the database from within VS, the account that you use to log on to your Windows machine needs at least read and write permissions to the folder where the database resides. If you are logged on as an Administrator, there's a fair chance this is already the case.

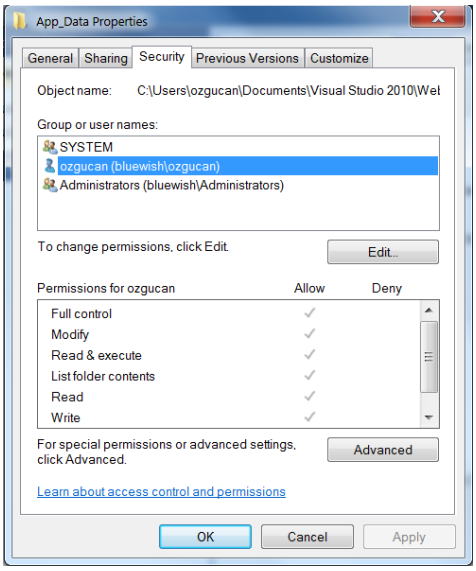
1. For this exercise you need a brand new web site, which you can create by choosing File ➪ New Web Site (or File ➪ New ➪ Web Site) in VS and then choose the ASP.NET Web Site template.



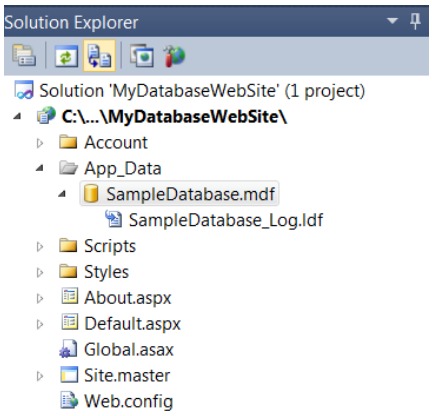

2. After you have created the new web site, ensure that your account has sufficient permissions to write to its `App_Data` folder. Because VS, and thus the built-in web server, runs under the account that you use to log in to Windows, you need to make sure that your account has the correct permissions. To this end, open Windows Explorer (not VS's Solution Explorer), and locate the folder:

```
C:\Users\ozgucan\Documents\Visual Studio 2010\WebSites\MyDatabaseWebSite
```

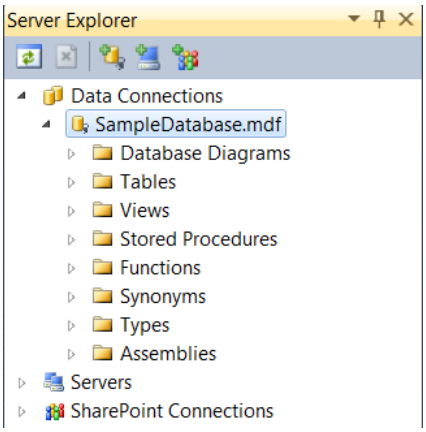
Right-click the App_Data folder and choose Properties. Switch to the Security tab, and ensure that your account (or a group you have been assigned to) has at least the Modify permission.




3. Select the file SampleDatabase.mdf using Windows Explorer. Arrange VS and the Windows Explorer side by side and then drag the file from the Windows Explorer into the App_Data folder of your web site in VS. Remember, if drag and drop doesn't work, you can accomplish the same thing using copy and paste. This .mdf file is the actual database and contains tables, records, and so on. When you start working with the database, you may also see an .ldf file appear in the App_Data folder on disk. This file is used by SQL Server to keep track of changes made to the database.



4. Double-click the database file in the Solution Explorer in VS. Doing so opens the database in the Database Explorer (called the Server Explorer in the commercial versions of Visual Studio).

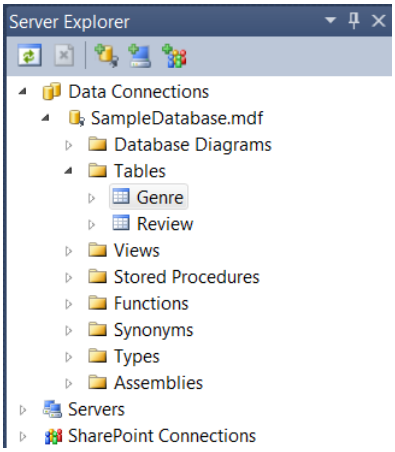



5. You can now expand the connected database to access its objects, such as the tables and columns it contains.

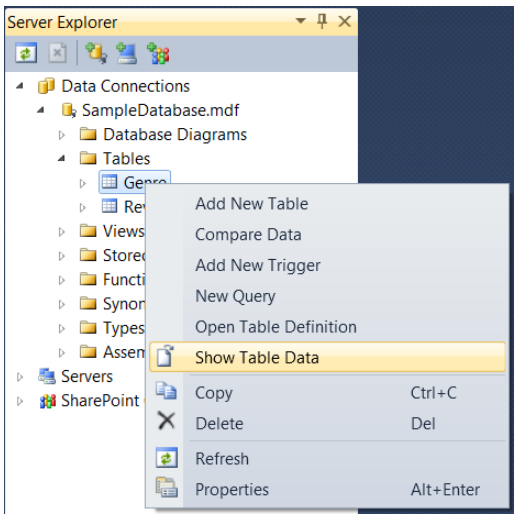
**Selecting Data From the Sample Database**

In this exercise you use the database that you connected to in an earlier exercise.

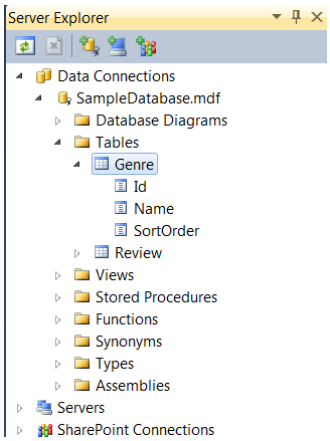
1. Open the Database Explorer (or the Server Explorer in the paid versions of Visual Studio) by choosing View ➪ Database Explorer. Locate the Data Connection that you added earlier, expand it, and then expand the `Tables` node. You should see two tables, `Genre` and `Review`.



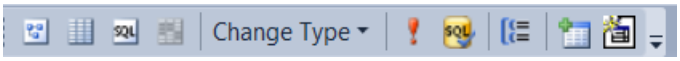

2. Right-click the `Genre` table and choose `Show Table Data`.



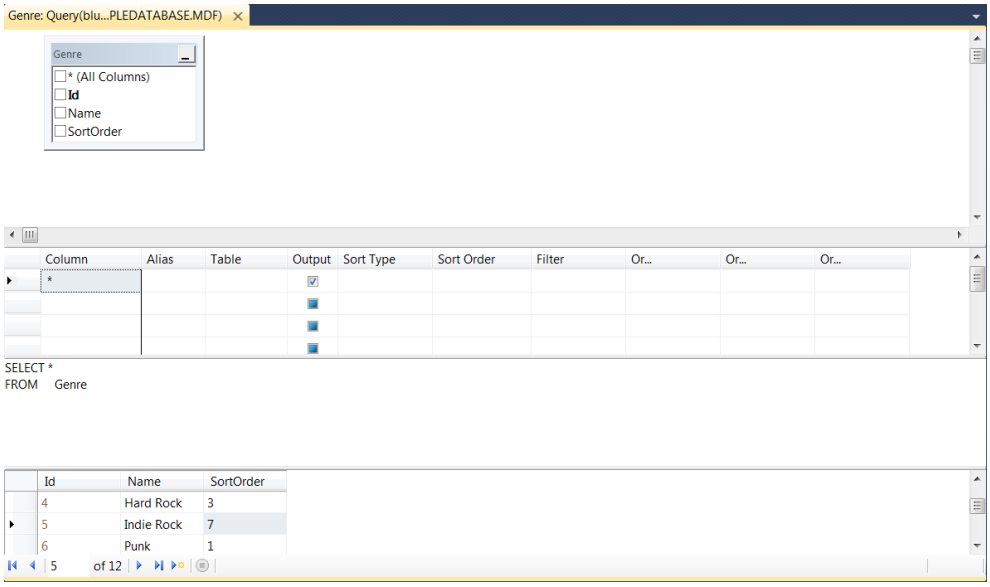

In the Document Window you should now see a list with all the available genres in the `Genre` table.




**Note** that this is not just a list with all the records in the `Genre` table. It's actually the result of a SQL `SELECT` query that is executed when you open the window. To see the query behind this list, ensure that the Query Designer toolbar is displayed onscreen. If the toolbar isn't visible, right-click an existing toolbar and choose Query Designer.




On this toolbar, click the `Diagram` pane, the `Criteria` pane, and the `SQL` pane buttons to open their respective windows. The first four buttons on the toolbar should now be in a pressed state and the Document Window is split in four regions, with each region corresponding to one of the buttons on the toolbar. The figure shows the entire Document Window with the four panes.

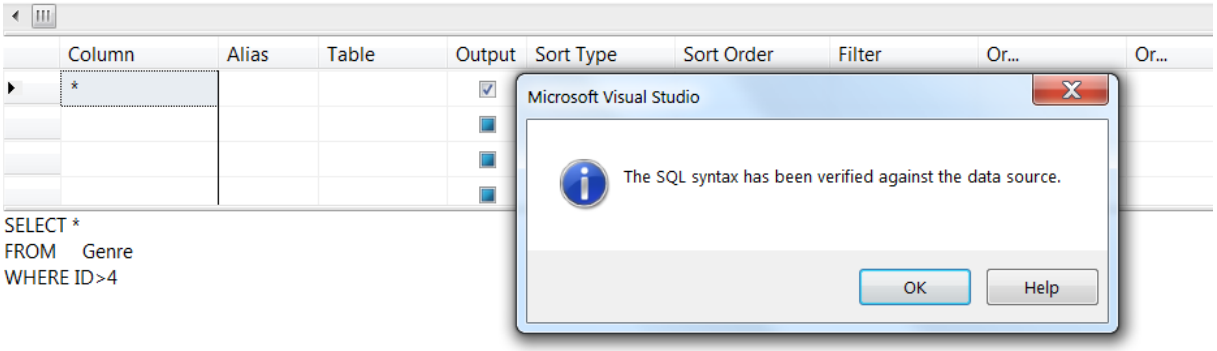


The SQL pane displays the SQL statement that is used to retrieve the genres that are displayed in the `Results` pane. In this case, the SQL statement reads `SELECT * FROM Genre` to retrieve all columns and records from the table, but you can easily change that.

3. In the SQL pane, position your cursor right after the word `Genre`, press Enter once, and then type `WHERE Id > 4`. Your complete SQL statement should end up like this:

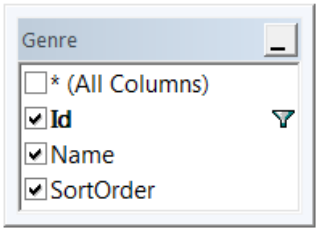
```
SELECT *
FROM    Genre
WHERE ID>4
```

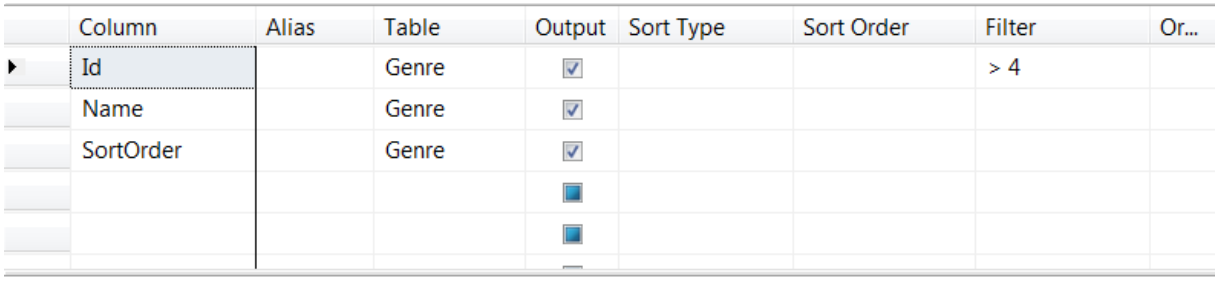
4. To make sure the SQL statement is valid, click the `Verify SQL Syntax` button on the toolbar and fix any errors your SQL statement may contain.



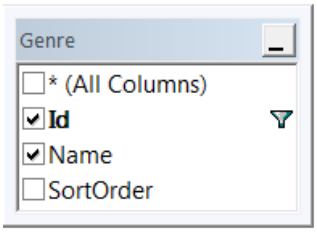

Next, click the `Execute SQL` button (the one with the red exclamation mark on it) or press Ctrl+R. In both cases, the SQL statement is executed and the Results pane is updated to show all genres with an `ID` larger than `4`.

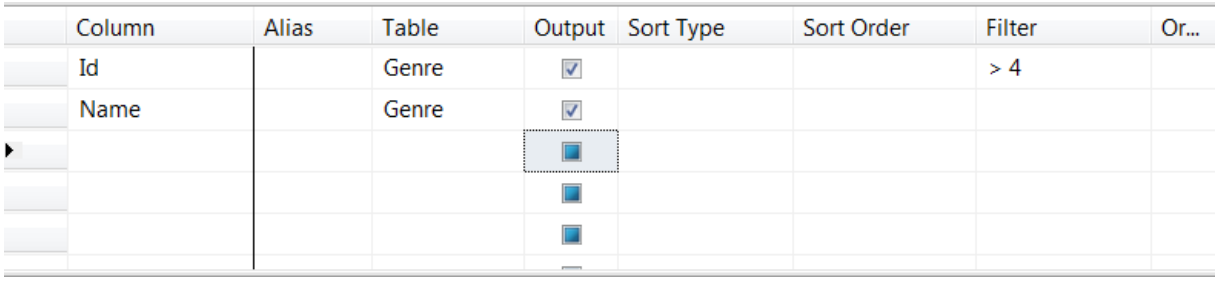
5. In addition to showing the results, VS also changed your query. Instead of `SELECT *`, it has listed each column in your table explicitly. Now take a look at the `Diagram` pane — the top part of the dialog box in the figure that shows your entire table. In the `Diagram` pane you can check and uncheck column names to determine whether they end up in the query.



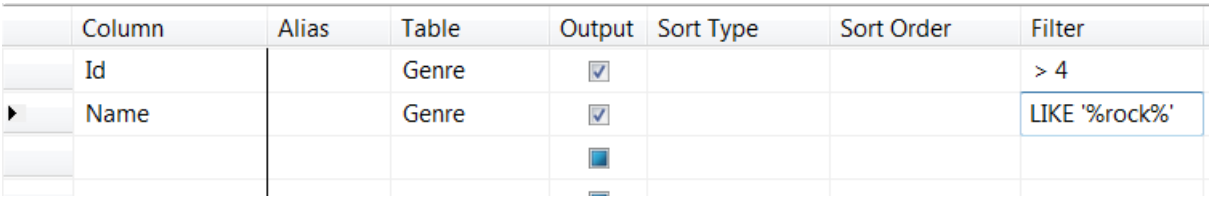




Deselect the `SortOrder` column (don't accidentally change the check mark of the `Output` column in the `Criteria` pane instead). **Note** that it also gets removed from the `Criteria` pane and the SQL statement in the SQL pane.



6. Take a look at the `Criteria` pane. It shows the two columns you are selecting. In the `Filter` column it shows the expression that filters all genres with an `ID` larger than `4`. In this pane you can modify the query without manually writing a lot of code. To see how you can apply an additional filter, type `LIKE '%rock%'` in the `Filter` cell for the `Name` row. This limits the results to all genres that contain the word `rock` and that have an `ID` that is larger than `4`.

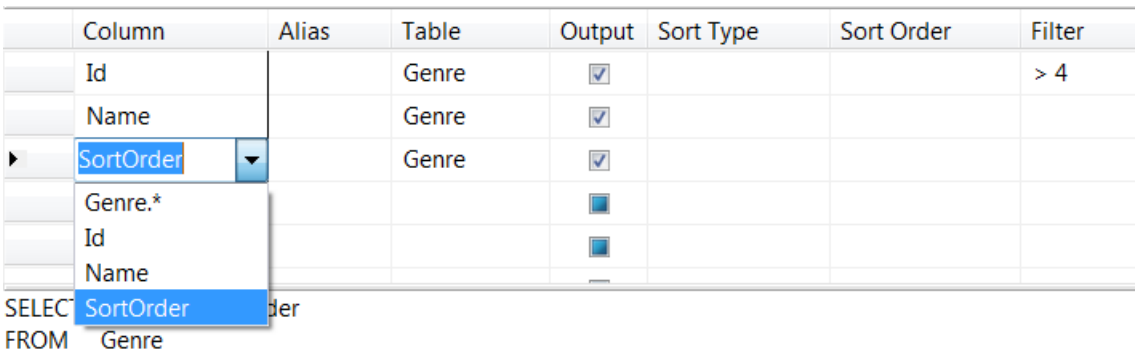


| Column | Alias | Table | Output | Sort Type | Sort Order | Filter |
|---|---|---|---|---|---|---|
| Id | | Genre | ☑ | | | > 4 |
| Name | | Genre | ☑ | | | LIKE '%rock%' |
| | | | ■ | | | |

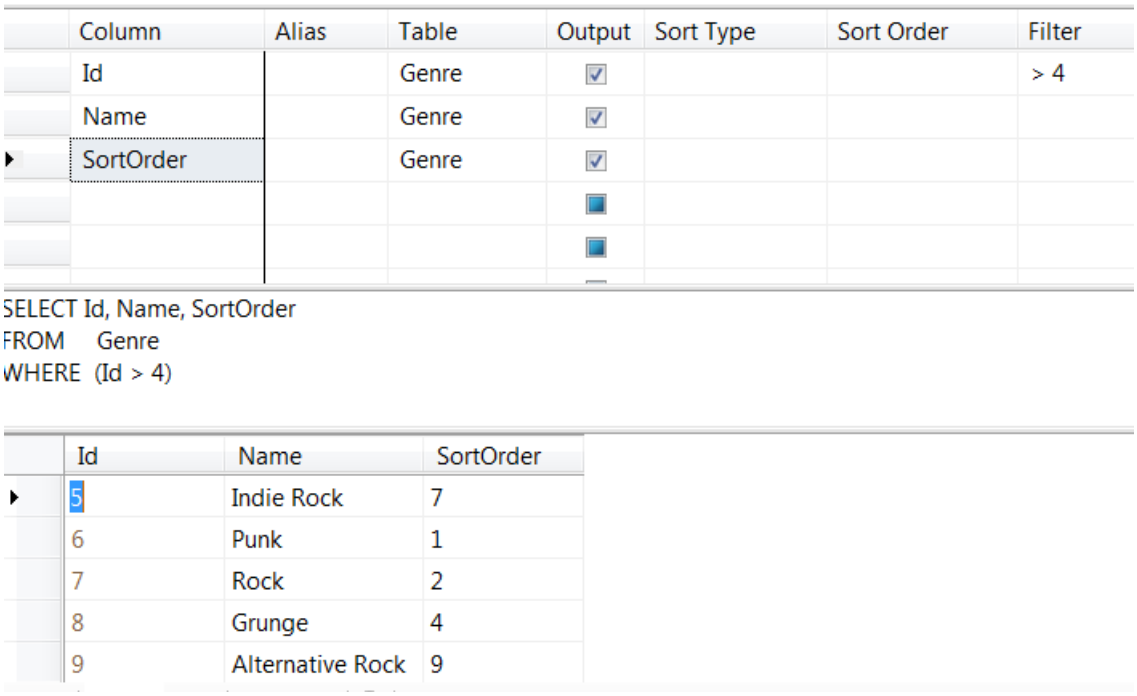
If you press Ctrl+R again the `Results` pane is updated to reflect the change in the query.

| Id | Name |
|---|---|
| 5 | Indie Rock |
| 7 | Rock |
| 9 | Alternative Rock |
| *NULL* | *NULL* |

7. To determine the sort order, you can use the `Sort Type` column. You can do this for visible columns (for example, those that have their `Output` check box checked and end up in the final result set) but also for other columns. To order by the `SortOrder` column, click the cell under `Name` once. It changes and now shows a drop-down list instead. Choose `SortOrder` from the drop-down list.



When you click or tab away from the field, VS places a check mark in the `Output` column.

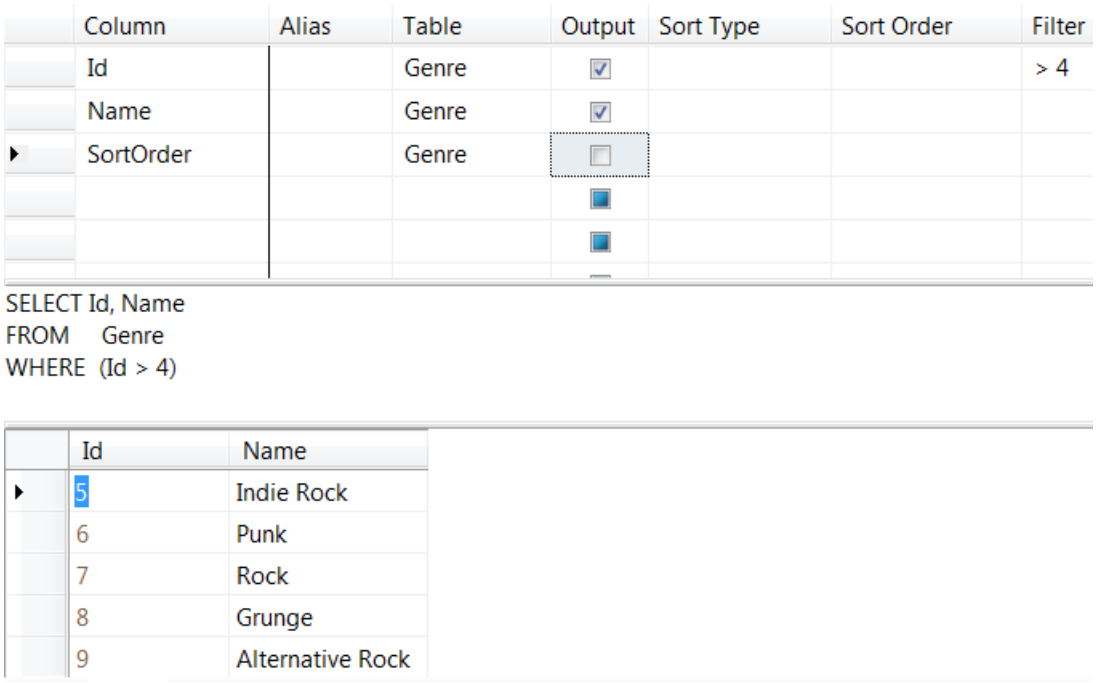


| Column | Alias | Table | Output | Sort Type | Sort Order | Filter |
|---|---|---|---|---|---|---|
| Id | | Genre | ☑ | | | > 4 |
| Name | | Genre | ☑ | | | |
| SortOrder | | Genre | ☑ | | | |

```
SELECT Id, Name, SortOrder
FROM   Genre
WHERE  (Id > 4)
```

| Id | Name | SortOrder |
|---|---|---|
| 5 | Indie Rock | 7 |
| 6 | Punk | 1 |
| 7 | Rock | 2 |
| 8 | Grunge | 4 |
| 9 | Alternative Rock | 9 |

You can click that check mark to remove the column again from the output so it remains available for ordering and filtering, but won't show up in the query results. However, for this exercise it's okay to leave the column selected.

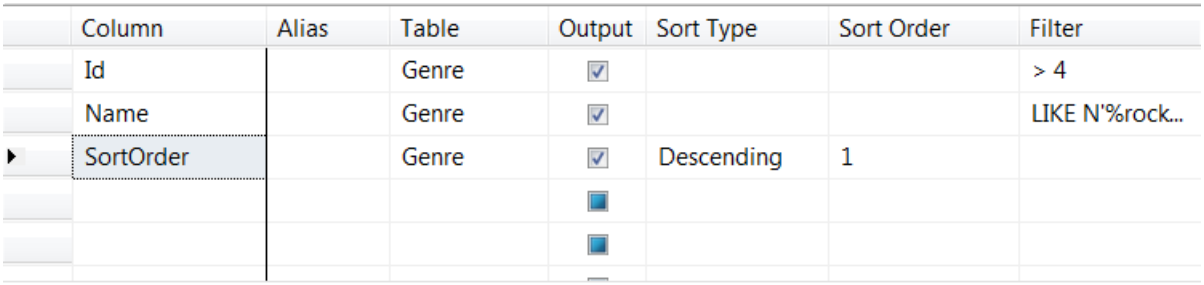


| Column | Alias | Table | Output | Sort Type | Sort Order | Filter |
|---|---|---|---|---|---|---|
| Id | | Genre | ☑ | | | > 4 |
| Name | | Genre | ☑ | | | |
| SortOrder | | Genre | ☐ | | | |

```
SELECT Id, Name
FROM   Genre
WHERE  (Id > 4)
```

| Id | Name |
|---|---|
| 5 | Indie Rock |
| 6 | Punk |
| 7 | Rock |
| 8 | Grunge |
| 9 | Alternative Rock |

8. In the `Sort Type` column choose `Descending` from the drop-down list for the `SortOrder`. Your final `Criteria` pane now looks like :



| | Column | Alias | Table | Output | Sort Type | Sort Order | Filter |
|---|---|---|---|---|---|---|---|
| | Id | | Genre | ☑ | | | > 4 |
| | Name | | Genre | ☑ | | | LIKE N'%rock... |
| ▶ | SortOrder | | Genre | ☑ | Descending | 1 | |
| | | | | ■ | | | |
| | | | | ■ | | | |

While you make your changes using the `Diagram` and `Criteria` panes, VS continuously updates the SQL pane. Your final SQL statement should now include the extra `WHERE` clause and the `ORDER BY` statement:

```
SELECT Id, Name, SortOrder
FROM    Genre
WHERE  (Id > 4) AND (Name LIKE '%rock%')
ORDER BY SortOrder DESC
```

9. Press Ctrl+R again (or click the `Execute SQL` button on the toolbar) and the `Results` pane shows the records from the `Genre` table that match your criteria:

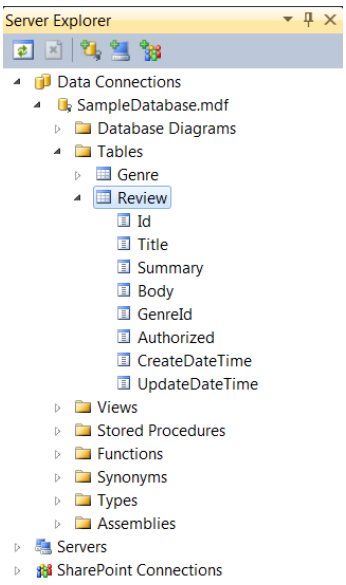
| | Id | Name | SortOrder |
|---|---|---|---|
| ▶ | 9 | Alternative Rock | 9 |
| | 5 | Indie Rock | 7 |
| | 7 | Rock | 2 |
| * | *NULL* | *NULL* | *NULL* |

**Note** that the records are now sorted in descending order based on the `SortOrder` column.

**Joining Data**

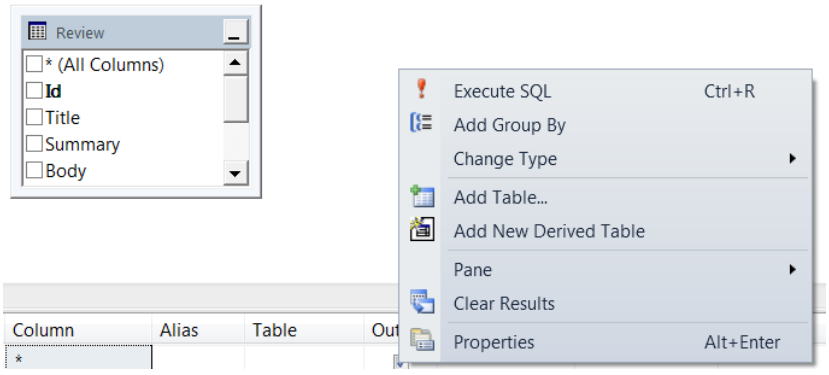
To join data from two tables, you need to write a `JOIN` statement in your code. To help you write the code, VS adds a `JOIN` for you whenever you add a table to the `Diagram` pane. However, sometimes this `JOIN` is not correct, so you'll need to check the code to see if it's okay.

1. Still in your test site, on the Database Explorer (or Server Explorer), right-click the `Review` table and choose `Show Table Data`. You'll see all the reviews in the table appear.

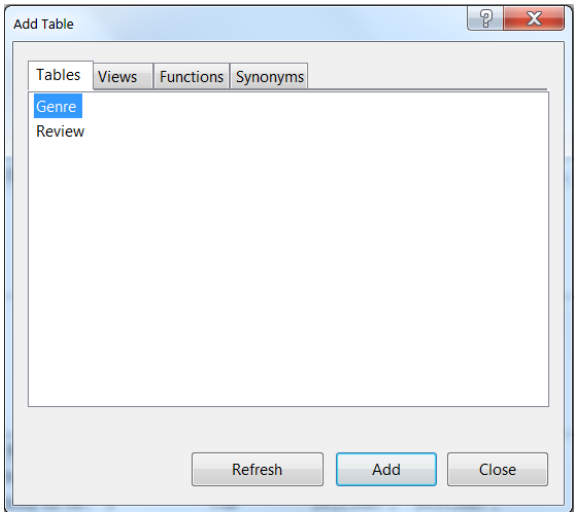


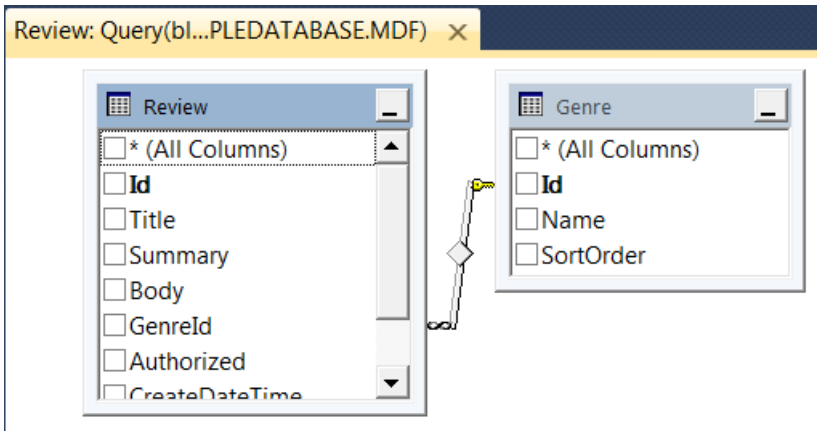

   Next, enable the `Diagram`, `Criteria`, and `SQL` panes by clicking their respective buttons on the Query Designer toolbar.

2. Right-click an open spot of the `Diagram` pane next to the `Review` table and choose Add Table. Alternatively, choose Query Designer ➪ Add Table from the main menu.



3. In the dialog box that opens, click the `Genre` table and then click the Add button. Finally, click `Close`.



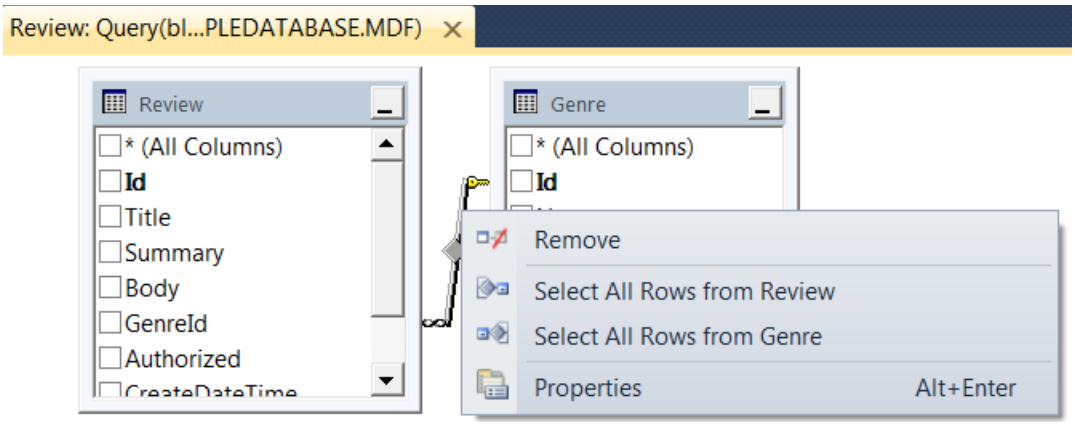




4. The SQL statement that VS generated looks like this:

```
SELECT *
FROM    Review
INNER JOIN Genre ON Review.GenreId = Genre.Id
```

VS correctly detected the relationship defined in the database between the `GenreId` column of the `Review` table and the `Id` column of the `Genre` table, and applied the correct `JOIN` for you.

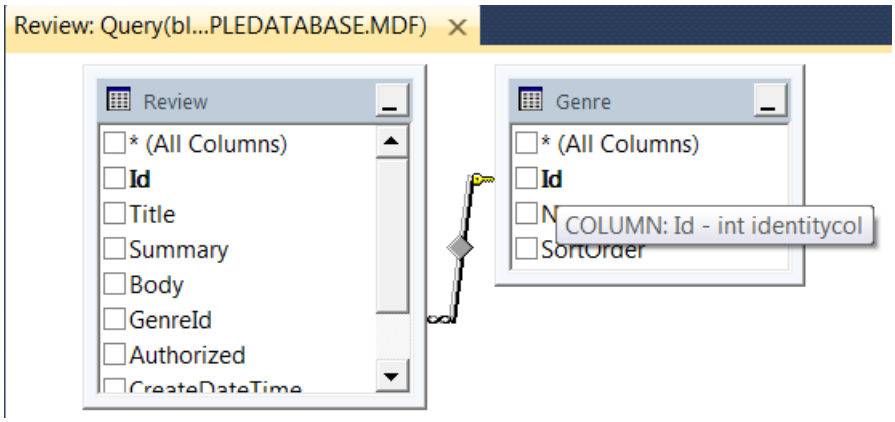
5. To see how you can create `JOIN`s yourself without writing code directly, you'll manually recreate the `JOIN`. First, right-click the line that is drawn between the two tables in the `Diagram` pane and choose `Remove`.



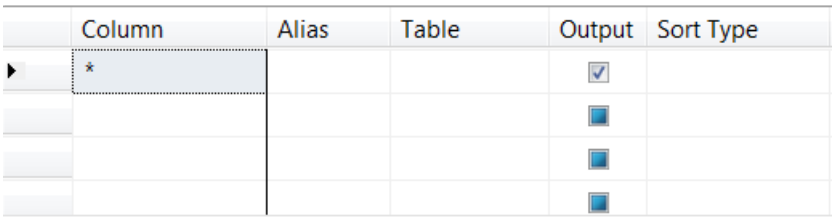

The SQL statement now contains a `CROSS JOIN`.

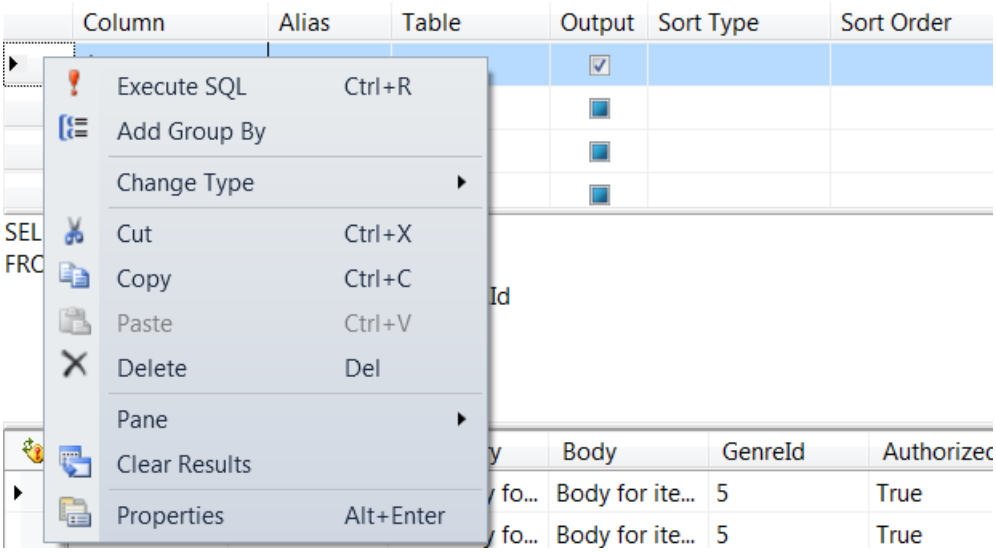
```
SELECT *
FROM    Review
CROSS JOIN  Genre
```

6. Next, click the `GenreId` column of the `Review` table in the `Diagram` pane once and drag it onto the `Id` column of the `Genre` table. As soon as you release the mouse, VS creates a new `INNER JOIN` in the `SQL` pane for you with the exact same code as you saw earlier.

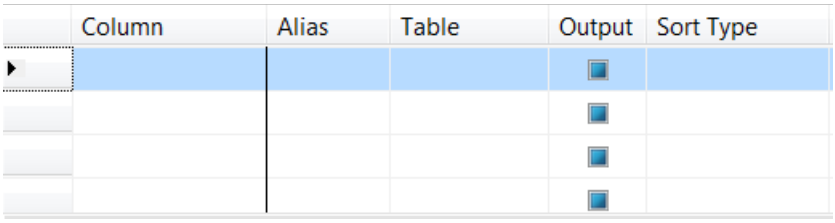


7. In the `Criteria` pane, click the left margin of the first row that contains the asterisk (`*`) symbol to select the entire row and then press the `Delete` key or right-click the left margin and choose `Delete`.

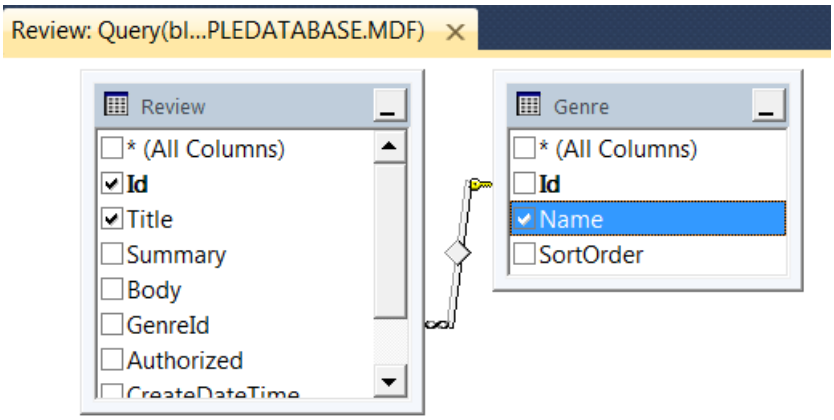


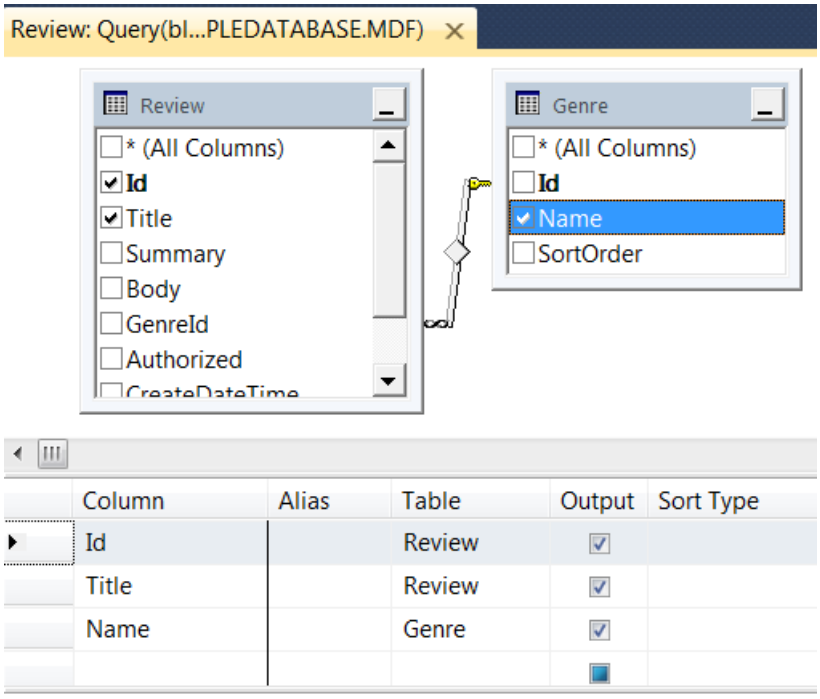

This removes the asterisk from the SQL statement. Alternatively, you can delete the asterisk from the `SQL` pane directly.




8. In the `Diagram` pane place a check mark in front of the `Id` and `Title` columns of the `Review` table and in front of the `Name` column of the `Genre` table.



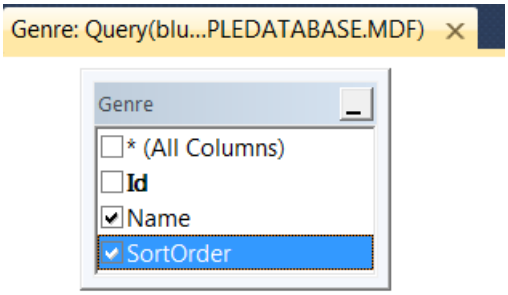

9. Finally, press Ctrl+R to execute the query. Your Document Window should show the results of the query at the bottom of the screen in the `Results` pane.




**Working with Data in the Sample Database**

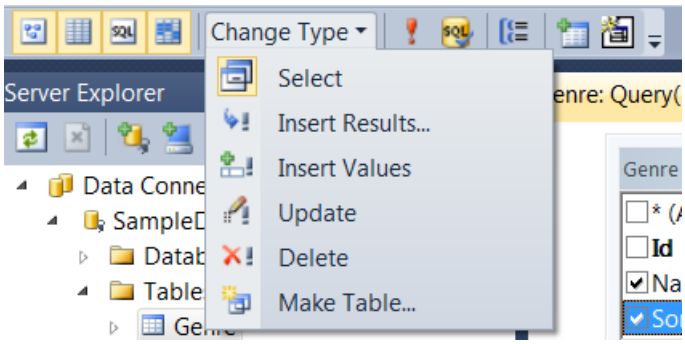
In this exercise, you see how to create a new record in the `Genre` table, select it again to find out its new `ID`, update it using the `UPDATE` statement, and finally delete the genre from the database. Although the examples themselves may seem pretty trivial, they are at the core of how SQL works. If you understand the examples from this section, you'll be able to work with the remaining SQL statements in the future.

1. Open the Database Explorer window in your temporary test site and locate the `Genre` table in the database. Right-click it and choose `Show Table Data`. If the table was already open with an old query, you need to close it first by pressing Ctrl+F4. This gets rid of the existing SQL statement.
2. Click the first three buttons on the Query Designer toolbar (`Diagram`, `Criteria`, and `SQL` pane) to open up their respective panes.
3. In the `Diagram` pane, check the columns `Name` and `SortOrder`. Make sure you leave `Id` unchecked.




Because the `Id` column gets an auto-generated value from the database, you cannot supply an explicit value for it in an `INSERT` statement.

4. On the Query Designer toolbar click the `Change Type` button and choose the third option: `Insert Values`.



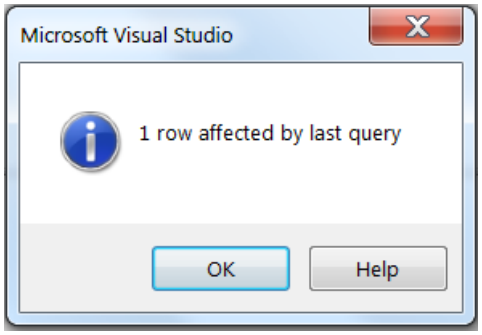

The query in the SQL pane is updated and now contains a template for the `INSERT` statement:

```
INSERT INTO Genre
            (Name, SortOrder)
VALUES (,)
```

5. Between the parentheses for the `VALUES`, enter a name (between apostrophes) and a sort order for your genre separated by a comma:

```
VALUES ('Folk',15)
```

6. Press Ctrl+R to execute the query. You should get a dialog box that tells you that your action caused one row to be affected.




7. Click `OK` to dismiss the dialog box.

8. Clear out the entire SQL statement from the `SQL` pane (you can use Ctrl+A to select the entire SQL statement and then press the Delete key to delete it) and replace it with this code, which selects all the genres and sorts them in descending order:

```
SELECT Id, Name
FROM Genre
ORDER BY Id DESC
```

9. Press Ctrl+R to execute this `SELECT` statement. The `Results` pane shows a list of genres with the one you just inserted at the top of the list. **Note** the `ID` of the newly inserted record. It should be `13` if you haven't inserted any record before although it's okay if you have a different `ID`.

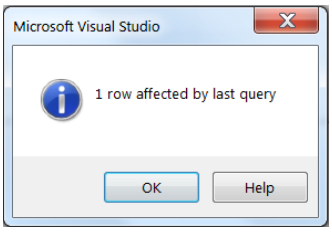
| | Id | Name |
|---|---|---|
| ▶ | 13 | Folk |
| | 12 | Techno |
| | 11 | Industrial |
| | 10 | Reggae |
| | 9 | Alternative ... |

10. Click the `Change Type` button on the toolbar again, this time choosing `Update`. Complete the SQL statement that VS created for you so it looks like this:

```
UPDATE Genre
SET       Name ='British Folk',
          SortOrder=5
WHERE Id=13
```

Don't forget to replace the number `13` in the SQL statement with the ID you determined in step 9.

11. Press Ctrl+R again to execute the query and you'll get a dialog box informing you that one record has been modified.



12. Once again, clear the SQL pane and then enter and execute the following query by pressing Ctrl+R:

```
SELECT Id, Name
FROM Genre
WHERE Id=13
```

Replace the Id in the WHERE clause with the ID of the record you determined in step 9. You should see the updated record appear.

| | Id | Name |
|---|---|---|
| ▶ | 13 | British Folk |
| * | NULL | NULL |

13. On the Query Designer toolbar, click the Change Type button and choose Delete. VS changes the SQL statement so it is now set up to delete the record with an ID of 13:

```
DELETE FROM Genre
WHERE  (Id = 13)
```

14. Press Ctrl+R to execute the query and delete the record from the database. Click OK to dismiss the confirmation dialog.
15. To confirm that the record is really deleted, click the Change Type button once more and choose Select. Then choose one or more columns of the Genre table in the Diagram pane and press Ctrl+R again. You'll see that this time no records are returned, confirming the newly inserted genre has indeed been deleted from the database.

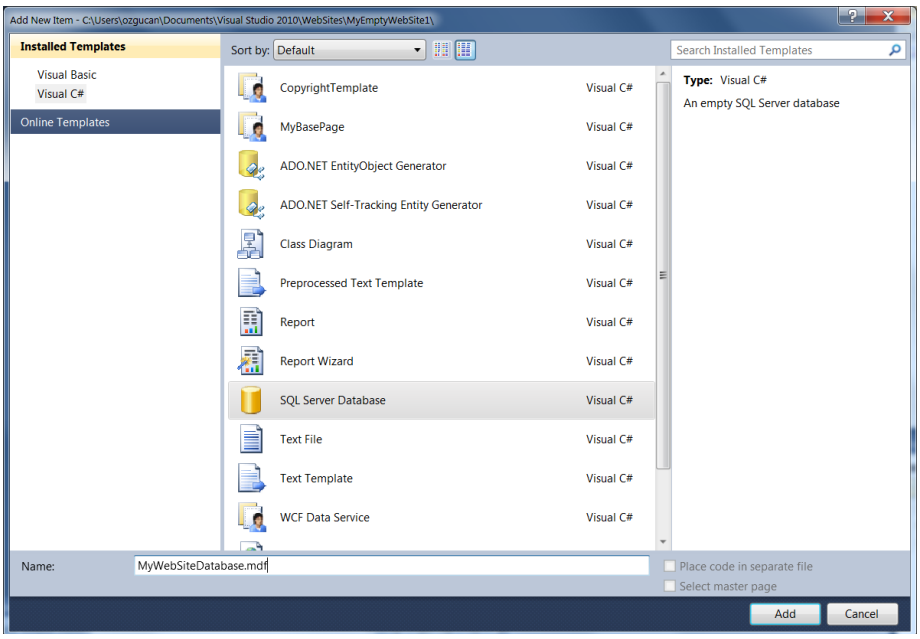
```
SELECT Id, Name
FROM    Genre
WHERE  (Id = 13)
```

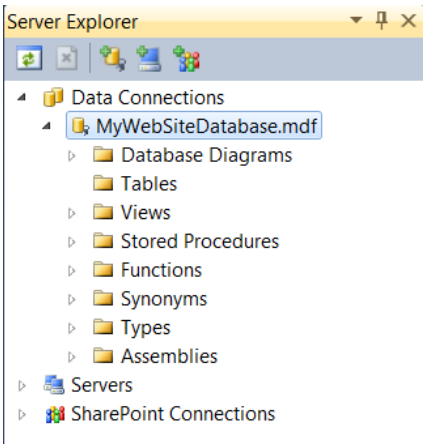
| | Id | Name |
|---|---|---|
| * | NULL | NULL |

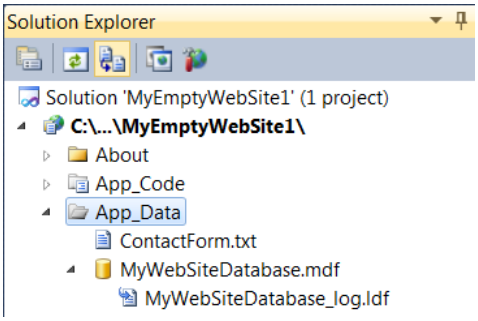
## Creating Tables in the Table Designer

In this exercise you add two tables to a new database that you add to the web site project you were working before. You should carry out the exercises in the web site you have been building in the past. You can close the test site you created at the beginning because you don't need it anymore. This exercise assumes you're creating tables for a local database (stored in the `App_Data` folder).

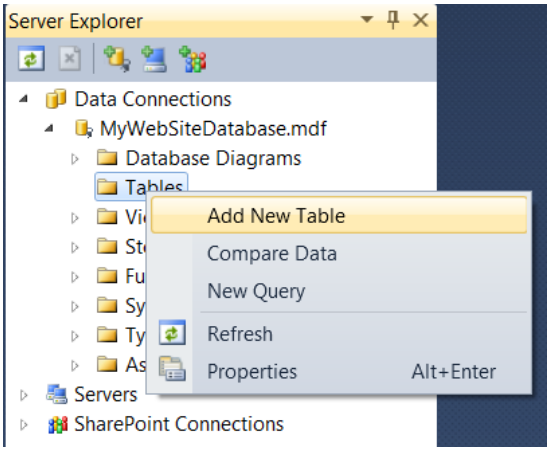
1. Open up the web site in VS, right-click the `App_Data` folder, and choose Add New Item. In the dialog box that follows, click SQL Server Database, type `MyWebSiteDatabase.mdf` as the name, and then click Add to add the database to your site. The Database Explorer (or Server Explorer) should open automatically showing you the new database. If it doesn't, double-click `MyWebSiteDatabase.mdf` in the Solution Explorer.



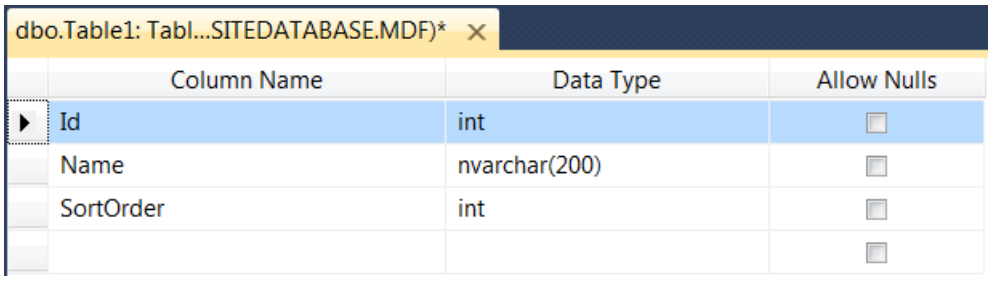



2. On the Database Explorer, right-click the `Tables` node and choose Add New Table.

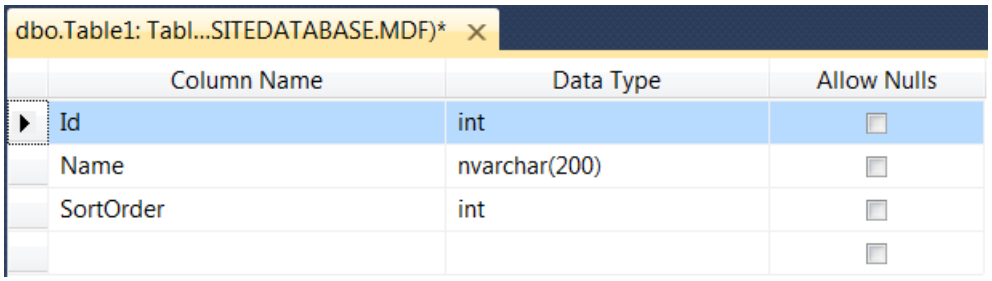



3. In the dialog box that follows, you can enter column names and data types that together make up the table definition. Create three columns for the `Id`, `Name`, and `SortOrder` of the `Genre` table so the dialog box ends up as shown in the figure.

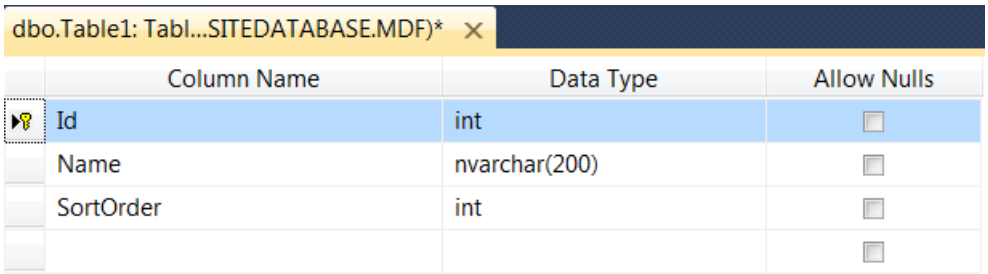



Make sure you clear the check box for all three items in the `Allow Nulls` column. This column determines if fields are optional or required. In the case of the `Genre` table, all three columns will be required, so you need to clear the `Allow Nulls` check box.

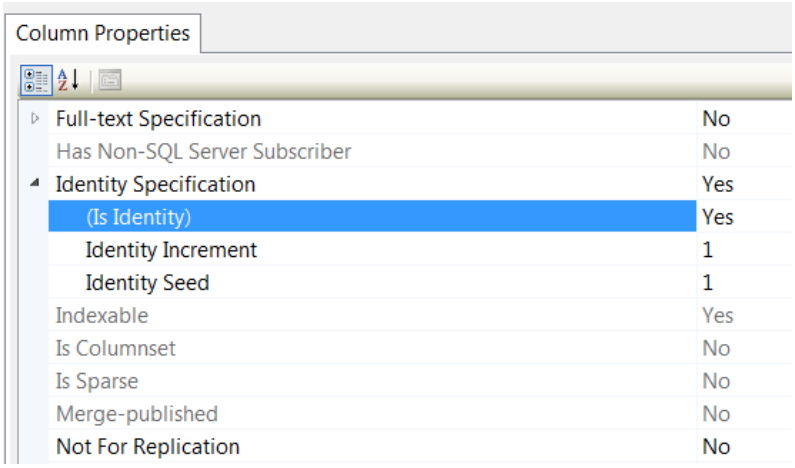
4. Next, select the entire row for the `Id` by clicking in the margin on the left (identified by the black arrow in the figure)



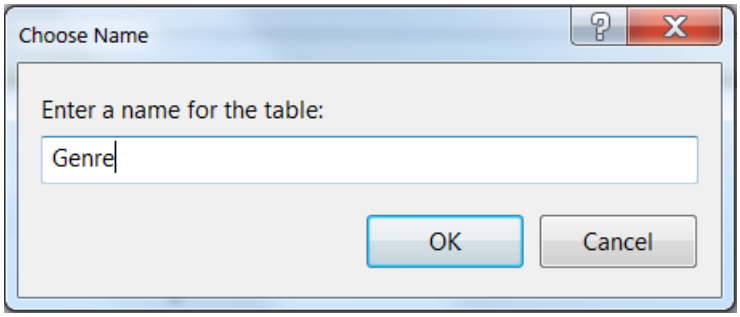

and then on the Table Designer toolbar, click the second button from the left (with the yellow key on it) to turn the `Id` column into a primary key.

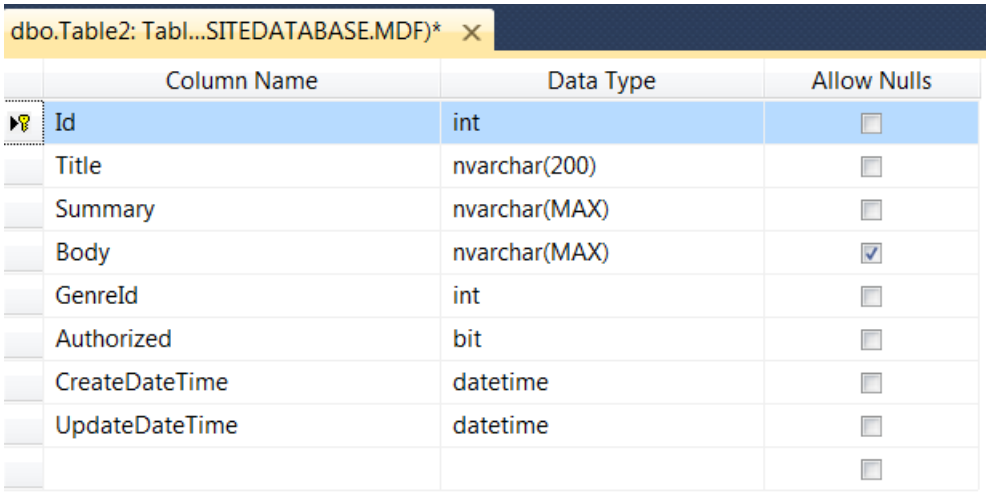
5. Below the table definition you see the Column Properties, a panel that looks similar to the Properties Grid in VS. With the `Id` column still selected, scroll down a bit on the Column Properties Grid until you see `Identity Specification`. Expand the item and then set (`Is Identity`) to `Yes`.



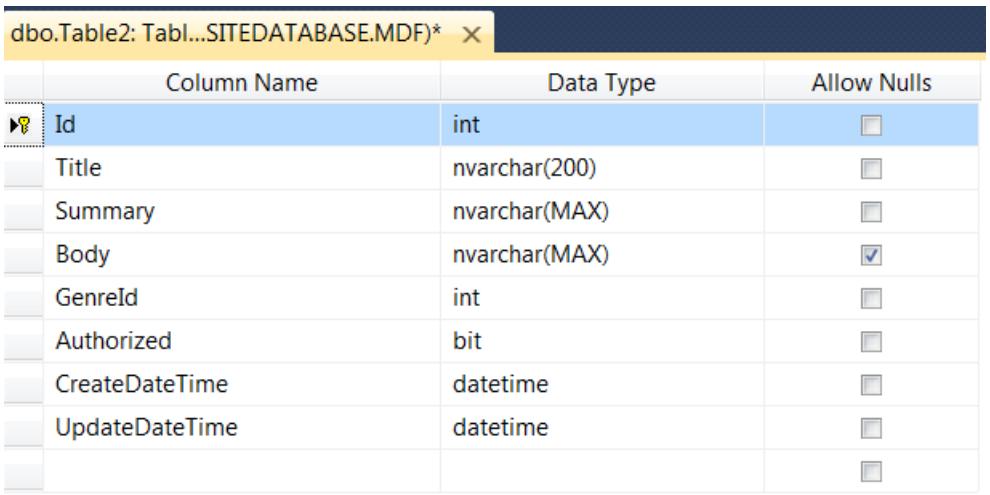

6. Press Ctrl+S to save your changes. A dialog box pops up that enables you to provide a name for the table. Type `Genre` as the name and click `OK` to apply your changes.



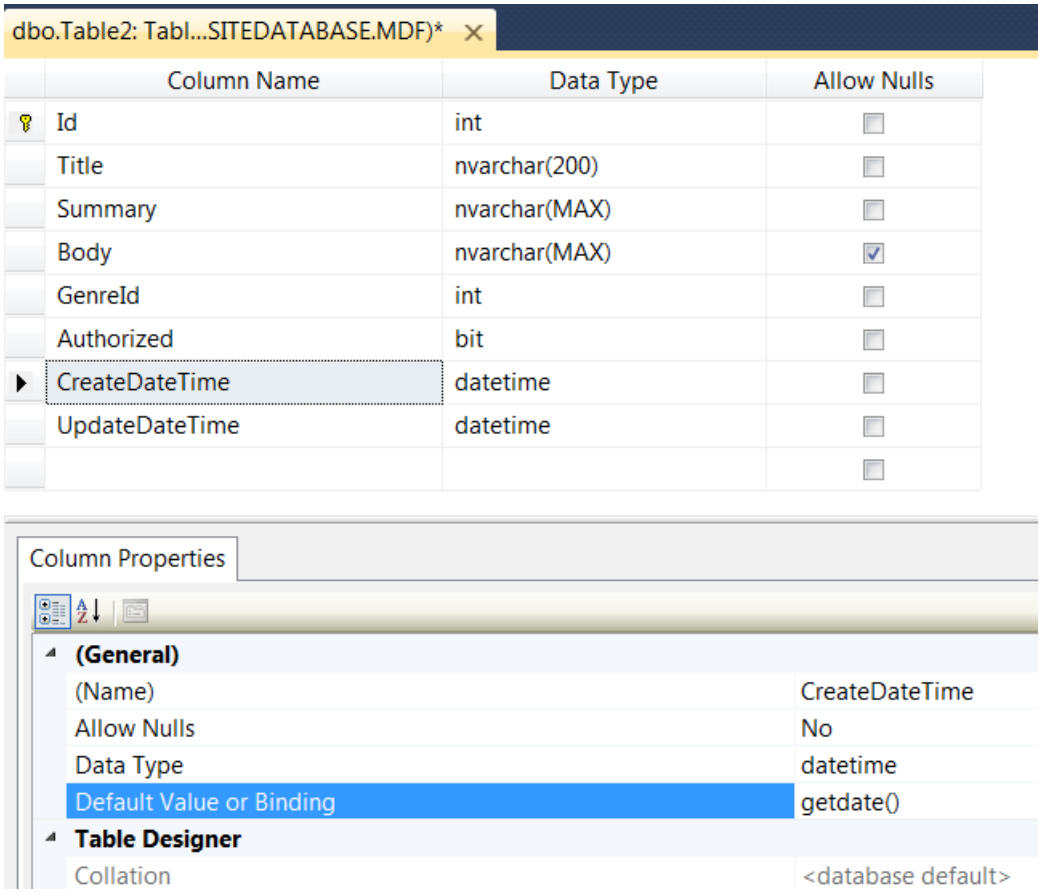
7. Create another table by following steps 2 and 3, but this time create a table with the following specifications to hold the CD and concert reviews for the web site.




8. Make the `Id` column the primary key again, and set its (`Is Identity`) property to `Yes` just as you did in steps 4 and 5.



9. Click the `CreateDateTime` column once and then on the Column Properties Grid, type `getdate()` in the field for the `Default Value` or `Binding` property.

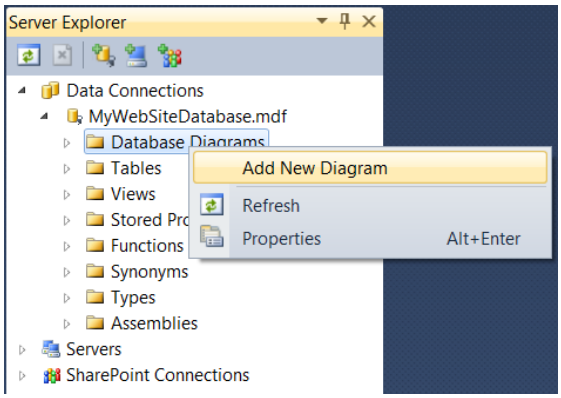



10. Repeat the preceding step for the `UpdateDateTime` column.
11. When you're done, press Ctrl+S to save the table and call it `Review`.

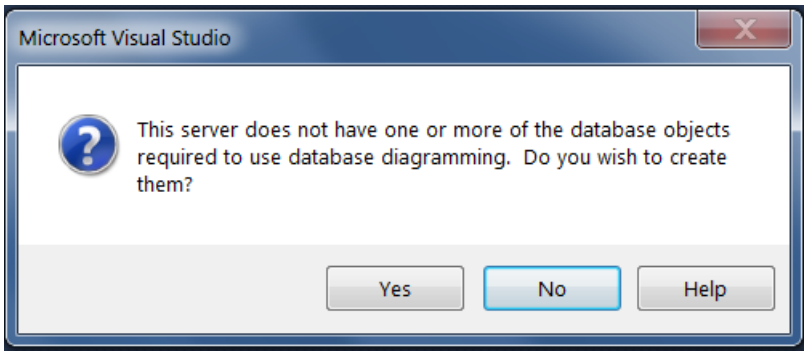
### Creating a Relationship Between Two Tables

Before you can visually add a relationship between two tables, you need to add a diagram to your database. A diagram is a visual tool that helps you understand and define your database. On the diagram, you can drag a column from one table to another to create the relationship. In this exercise, you create a relationship between the `Review` and `Genre` tables.

1. Open the Database Explorer again for the web site. Right-click the Database Diagrams element and click Add New Diagram.

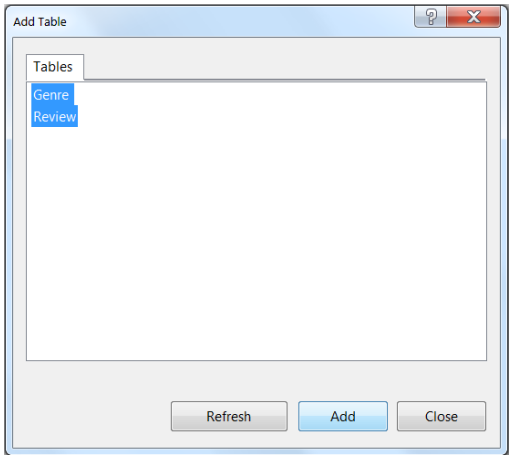



If this is the first time you are adding a diagram to the database, you may get a dialog box asking if you want VS to make you the owner of the database. Click `Yes` to proceed.

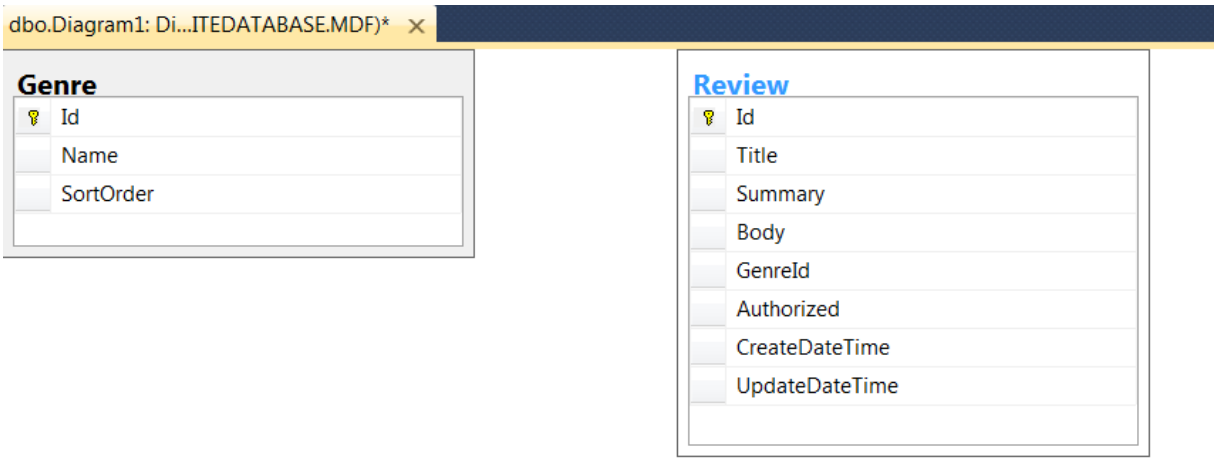



Don't worry if you don't get this prompt; things will work fine without it. The prompt may be followed by another that indicates that in order to work with diagrams, VS needs to create a few required objects. Again, click `Yes` to proceed.

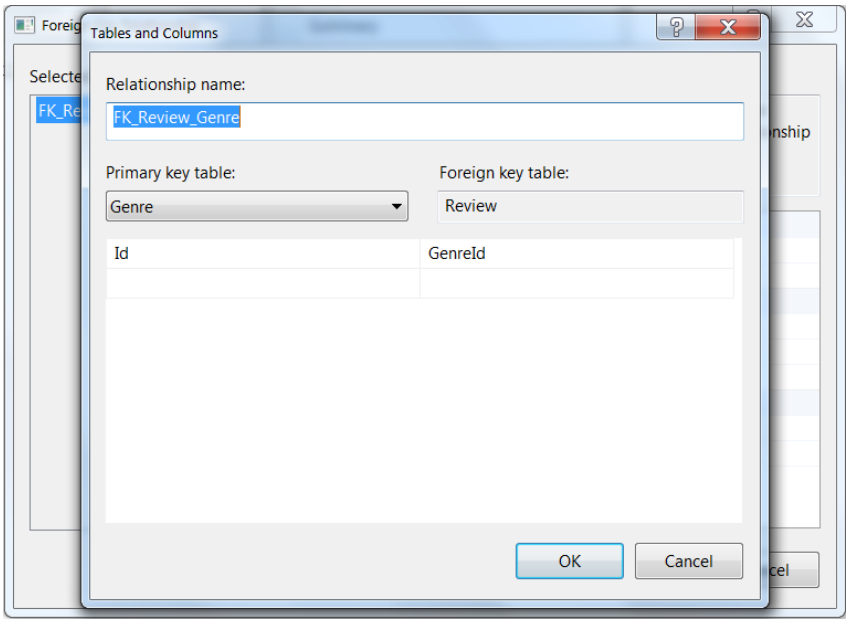
2. In the Add Table dialog box that follows, select both tables you created in the previous example (hold down the Ctrl key while you click each item), click `Add` to add the tables to the diagram, and then click `Close` to dismiss the Add Table dialog box.



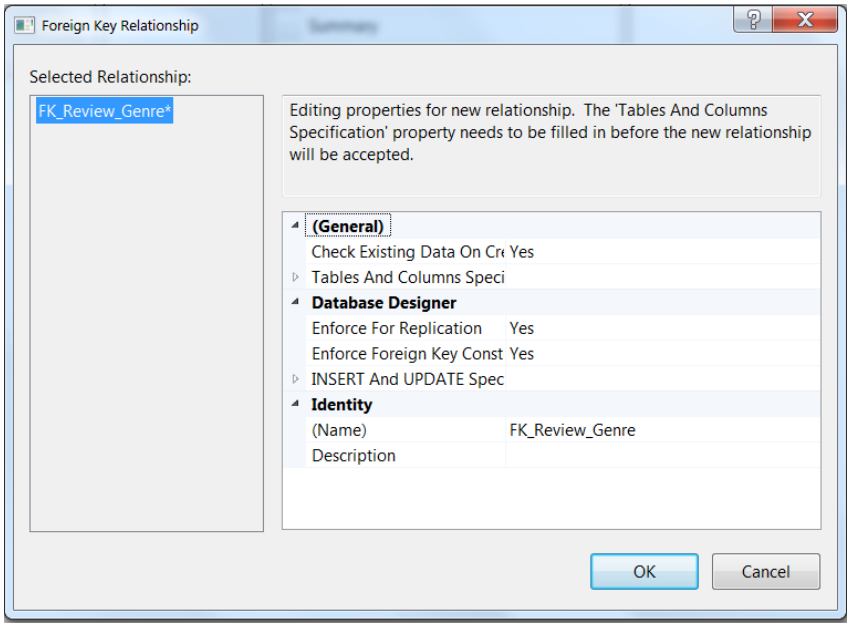

3. If necessary, arrange the tables in the diagram using drag and drop so they are positioned next to each other.




4. On the `Genre` table, click the left margin of the `Id` column (it should contain the yellow key to indicate this is the primary key of the table) and then drag it onto the `GenreId` column of the `Review` table and release your mouse.
5. Two dialog boxes pop up that enable you to customize the defaults for the relation.

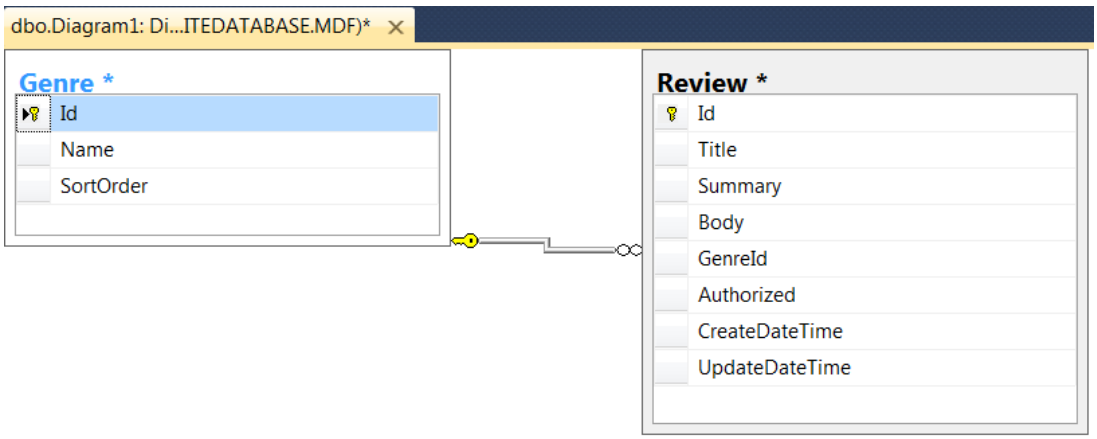



In the top-most window, confirm that `Id` is selected from `Genre` as the `Primary Key Table` and that `GenreId` is selected from `Review` as the `Foreign Key Table`. Click `OK` to dismiss the top window and confirm the columns that participate in the relationship.

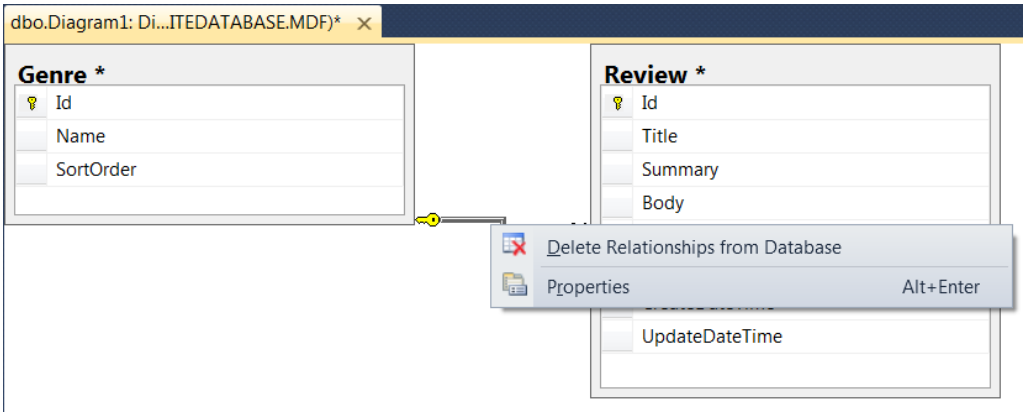



In the dialog box that remains, notice how `Enforce Foreign Key Constraint` is set to `Yes`. This property ensures that you cannot delete a record from the `Genre` table if it still has reviews attached to it. Click `OK` to dismiss this dialog box as well.

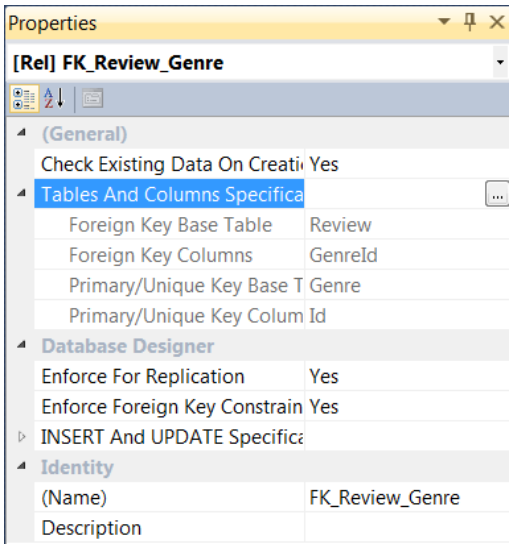
6. The diagram window should now show a line between the two tables. At the side of the `Genre` table, you should see a yellow key to indicate this table contains the primary key for the relationship. At the other end, you should see the infinity symbol to indicate that the `Review` table can have many records that use the same `GenreId`.



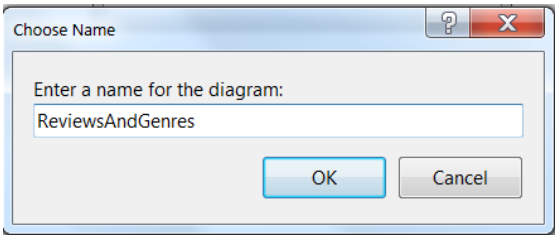

**Note** that the line between the two tables doesn't necessarily point to the correct columns. This can be confusing sometimes because you may think that other columns are actually related. To confirm the columns participating in the relationship, right-click the line between the two tables and choose `Properties`.



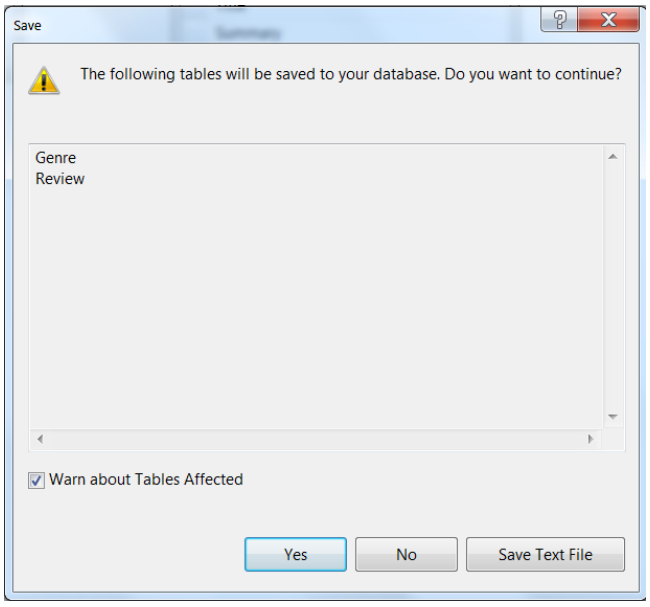

The `Table` and `Columns Specification` item shows which columns and tables participate in the relationship.




7. Press Ctrl+S to save the changes to the diagram. You can leave the name set to its default of `Diagram1` or you can enter a more descriptive name such as `ReviewsAndGenres` and click `OK`.




You'll get another warning that states that you are about to make changes to the `Review` and `Genre` tables. Click `Yes` to apply the changes.



8. Go back to the Database Explorer, right-click the `Genre` table, and choose `Show Table Data`. Enter a few different genres by typing a `Name` and a `SortOrder`. When you press Tab in the `SortOrder` field to tab away from the current row, the row is inserted in the database, and the `Id` column is filled with a unique, sequential number. You should end up with a list similar to the one shown in the figure.

Genre: Query(blu...SITEDATABASE.MDF)

| Id | Name | SortOrder |
|---|---|---|
| 4 | Pop | 12 |
| 5 | Jazz | 4 |
| 6 | Rock | 2 |
| 7 | Hard Rock | 7 |
| 8 | Classical | 6 |
| 9 | Blues | 1 |
| 10 | Chicago Jazz | 3 |
| *NULL* | *NULL* | *NULL* |

9. Open the `Review` table from the Database Explorer using the `Show Table Data` command and enter a few review records. For the `GenreId`, supply some of the new `IDs` you got when you inserted records in the `Genre` table. You can just make up the `Title`, `Summary`, and `Body` fields for now and set `Authorized` to `True`. Remember, you don't have to enter a value for the date columns. If you leave them out, the database will insert the default value for you. Notice that you can't insert a value in the `Id` column yourself. Because this column is an `Identity` field, the database supplies values for you automatically. If you get an error about missing values for the date columns, ensure that you entered a proper default value in the previous exercise. When you're done entering a row, click outside the row (on the new, empty row below it, for example) and press Ctrl+R to insert the row in the table. Your list of records should look similar to the figure, although your content for the columns, of course, may be different.

Review: Query(bl...SITEDATABASE.MDF) | Genre: Query(blu...SITEDATABASE.MDF)

| Id | Title | Summary | Body | GenreId | Authorized | CreateDate... | UpdateDat... |
|---|---|---|---|---|---|---|---|
| 1 | 21-Adele | Adele "the ... | *NULL* | 4 | True | 29.04.2013 ... | 29.04.2013 ... |
| 2 | Doo-Wops & Hooligans-Bruno... | Bruno Mars ... | *NULL* | 4 | True | 29.04.2013 ... | 29.04.2013 ... |
| 3 | Random Access Memories-Daft... | The fourth s... | *NULL* | 6 | True | 29.04.2013 ... | 29.04.2013 ... |
| 4 | Beethoven's 5 Secrets-The Pian... | Beethoven | *NULL* | 8 | True | 29.04.2013 ... | 29.04.2013 ... |
| *NULL* | *NULL* | *NULL* | *NULL* | *NULL* | *NULL* | *NULL* | *NULL* |

10. Right-click the `Genre` table again and choose `Show Table Data`. Click the SQL pane button on the Query Designer toolbar and then use the `Change Type` button on the same toolbar to create a `DELETE` query. Modify the query so it looks like this:

```
DELETE FROM Genre WHERE Id=8
```

This code will attempt to delete the `Classical` genre. However, because reviews are connected to it, the delete action should fail. Make sure that the `Id` in the `WHERE` clause matches one of the genre `ID`s you used in step 9 to link the reviews to. Press Ctrl+R to execute the query. Instead of deleting the record from the `Genre` table, VS now shows you the dialog box you see in the figure.