

|  |  |
| --- | --- |
|  | MS SQL Server Quick Guide 02: |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |



Quick Guides

This SQL Workshop looks to how we can develop a SQL Server Database with multiple related Tables and examines how we can Insert data, generate Views and Query data across related tables. We will start with a single table and demonstrate how we scale the solution into a multi related tables that meet the needs of a mobile app or website. We will examine how to insert data into related tables. How we can generate Views across related data. This series covers how we can generate Views across by utilising Joins for related Tables. Subqueries are discussed as well as group by constructs.

How to Build DB Tables and Relationships

- Music Example

Microsoft SQL Server is a Relational Database Management System (RDBMS)

developed by Microsoft. It is a highly scalable product that can be run on anything from

a single laptop, to a network of high-powered cloud servers, and anything in between.

This **SQL Quick Guide** demonstrates how wecan **create** a **database** with 3 related **tables, insert data, generate views** and perform some basic **queries** across related tables in **SQL Server Management Studio** ([**SSMS**](https://docs.microsoft.com/en-us/sql/ssms/quickstarts/ssms-connect-query-sql-server?view=sql-server-ver15)) GUI.

The SQL Server skills to acquire from the SQL Quick Guides are:

SQL Quick Guide 01 covers the following:

* Installing SQL Server and SSMS for home use
* How to login to SQL Server Management Studio SSMS
* How to Navigate SQL Server Management Studio SSMS
* Create a Database and a Table
* create a Table and replicate or amend the TSQL DDL Code
* add Constraints if required.
* insert data into the table.
* query the data by selecting records.

**SQL Quick Guide 02: covers the following:**

* **Create a Tables via SSMS GUI**
* **Create a Tables using SQL Script Code**
* **Create a Relationship between many Tables.**
* **Insert Data and Import Data**

SQL Quick Guide 03: covers the following:

* Query Designer
* Create a View and the View Designer
* Save Query Results to CSV File
* *Script a Database – separate guide!*

Content Overview:

[1. Introduction to the Music Case Study 3](#_Toc77350634)

[2. Creating a simple Music Database – 3 table solution in SQL Database 4](#_Toc77350635)

[3. Create a Database - music. 4](#_Toc77350636)

[3.1. Create a Table using the GUI – Artists table 8](#_Toc77350637)

[3.2. . Add Columns. 9](#_Toc77350638)

[3.3. Set an Identity Column 9](#_Toc77350639)

[3.4. Set a Primary Key 10](#_Toc77350640)

[3.5. Save the Table. 11](#_Toc77350641)

[3.6. Check that the Table was Created. 12](#_Toc77350642)

[3.7. Refresh the Tables Node 13](#_Toc77350643)

[4. Create a Table using the SQL script code – genre table 14](#_Toc77350644)

[4.1. Open a New Query Window 15](#_Toc77350645)

[4.2. Run the SQL Script 15](#_Toc77350646)

[4.3. Check that the Table was Created. 16](#_Toc77350647)

[4.4. Table Options 17](#_Toc77350648)

[5. Create a relationship between tables by using the GUI or SQL script. 17](#_Toc77350649)

[5.1. The Method 17](#_Toc77350650)

[5.2. Create a Relationship using SQL 18](#_Toc77350651)

[5.3. Some explanation on the SQL Script Code. 18](#_Toc77350652)

[5.4. What's a Foreign Key Constraint? 19](#_Toc77350653)

[5.5. Create a Relationship via the GUI 19](#_Toc77350654)

[5.6. Open the Child Table in the Table Designer 20](#_Toc77350655)

[5.7. Open the Foreign Key Relationships Dialog 21](#_Toc77350656)

[5.8. Add the Relationship 22](#_Toc77350657)

[5.9. Select Tables And Columns Specification 23](#_Toc77350658)

[5.10. The Tables and Columns Dialog Box 24](#_Toc77350659)

[5.11. The Relationship 25](#_Toc77350660)

[5.12. Saving The Relationship 25](#_Toc77350661)

[6. Insert data using a number of methods such as directly, copy/paste, import, use SQL, or use an external application. 26](#_Toc77350662)

[6.1. Direct Input 26](#_Toc77350663)

[6.2. Import Data 29](#_Toc77350664)

[6.3. Use SQL to Insert the Data 30](#_Toc77350665)

[6.4. Website/Application 32](#_Toc77350666)

[7. SQL Server Management Studio import wizard to help you import data from an external source. 33](#_Toc77350667)

[7.1. 1. Import a CSV File. 33](#_Toc77350668)

[7.2. 1. Import Data. 34](#_Toc77350669)

[7.3. Welcome to the SQL Server Import and Export Wizard. 35](#_Toc77350670)

[7.4. Choose a Data Source. 36](#_Toc77350671)

[7.5. Advanced Settings. 39](#_Toc77350672)

[7.6. Choose a Destination. 41](#_Toc77350673)

[7.7. Column Mappings. 44](#_Toc77350674)

[7.8. Review Data Type Mapping. 46](#_Toc77350675)

[7.9. Save and Run the Package. 47](#_Toc77350676)

[7.10. Complete the Wizard. 48](#_Toc77350677)

[7.11. Execution Report: Success or Failure. 49](#_Toc77350678)

[7.12. Check the Data. 53](#_Toc77350679)

# Introduction to the Music Case Study

We are only concerned with developing a backend SQL Server Database but for what?

Typically, the **SQL Server database** hold the data repository for a Mobile App, Website for integration into some Applications. In fact, **SQL Server** can integrate into many of the Development Tools and Technology since its conception in 1987.

What are the typical requests **TU Student** or a **Junior Database Develop** may have to deal with? Here are 3 typical request that came in during the same week to **SCEDT**. These will become **live Projects** or **ICAs** for our **UG** and **PG**. It is a great way to gain valuable **work experience** but also help you **develop** and **build** your **skills** to an **Industry standard**. Become fit for work within the Industry.

* 24.03.21: [**NHS – James Cook Hospital**](https://digital.nhs.uk/) request advice on setting up a bespoken Mobile App / Website to track progress and tasks on their Research Projects. The NHS – Research Task Manager App has been given to YR2, YR3 and PG students to develop.
* 24.03.21: [**DTN Academy Ltd**](https://dtnacademy.com/) request support in developing an eLearning Platform to deliver its teaching content. Given to YR2 to develop the front-end App and Backend Database.
* 25.03.21: [**Rolling Stones Magazine**](https://www.billboard.com/music/the-rolling-stones/chart-history/TLP) request support in developing a Mobile App for their top **500 Album List**. Given to YR1 to develop the SQL Server Database and YR2 to develop the full App.

All the above will require a Database to hold the repository of data to support and array of features and functions provided by the Mobile App and/or Website. The 3 systems may appear different, but you will be surprised how easy it is to develop a Database solution that services all of the above.

**Modules that SQL Server database support:**

UG YR2 – Mobile App Development based modules.

UG YR2 – Web Development based modules.

UG YR2 – Group Project.

UG YR3 – Final Year Project.

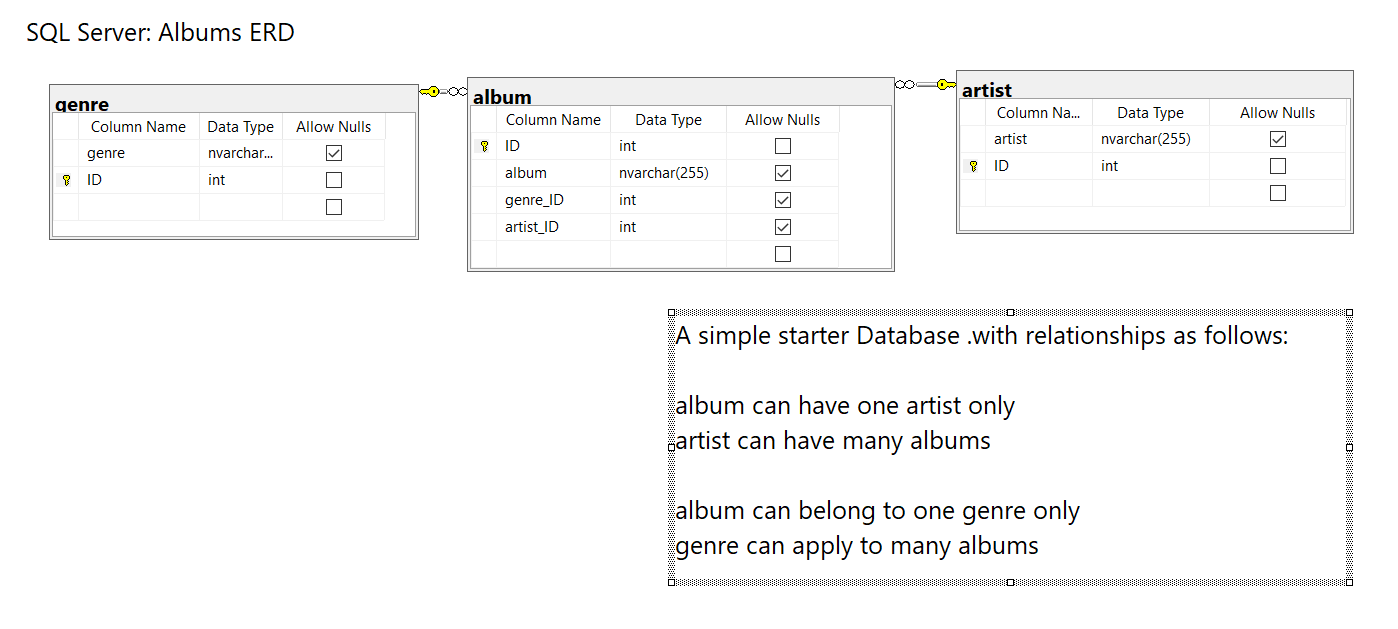
UG YR3 – Business Intelligence BI, Machine Learning MI or ML and Artificial Intelligence AI.

So, for the SQL Basic Guide we are going to develop a simple backend database solution for Rolling Stones Magazine.

Attached as part of this **SQL Server Guide** is a **YR2 Group Project Report** on **developing** a **MusicZone** website. Spend **15 minutes** scanning over the report and in particular note the sections on **Database Design** and the **Implemented Database**. The content is a collation of skills acquired from **several modules** and is your chance to **showcase** your acquired **skills** as **a junior developer**. You will be expecting to work as a Team/Group on **designing** and **building** a **Mobile App** and/or **Website** for **Industry**.

# Creating a simple Music Database – 3 table solution in SQL Database

**Case Study:**



The above is a typical ERD design for a solution for a simple Albums Database.

We have records of **Albums** in our model. Each **Album** must have an **Artist(s)** and an **Album** can belong to many **Genres**.

We therefore generate an **Artists** table since **Artist** can be associated with many **Albums**.

We therefore generate a **Genre** table since a **Genre** can be associated with many **Albums**.

We can then simply link or build a relationship between the keys associated with each of the tables.

# Create a Database - music.

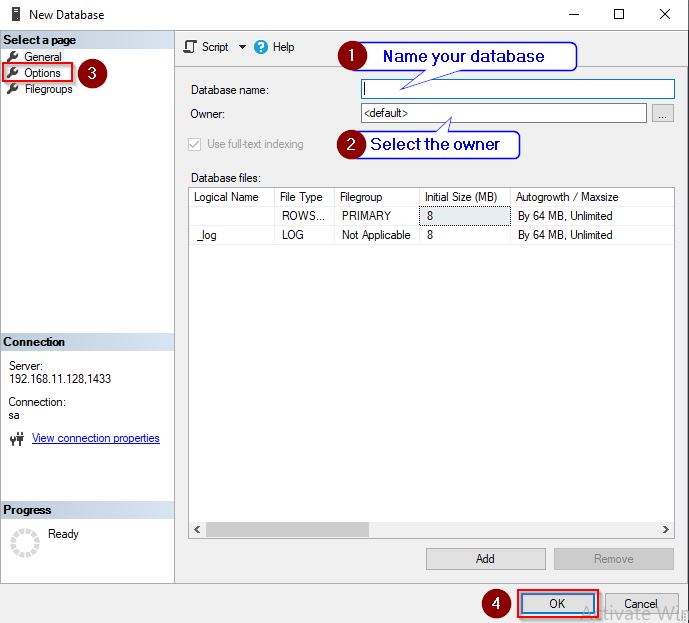
First make sure you have login to **SQL Server Management Studio** ([**SSMS**](https://docs.microsoft.com/en-us/sql/ssms/quickstarts/ssms-connect-query-sql-server?view=sql-server-ver15)) and created the Database **music.**

In this section, you will create a table in the **music** database by follow the instructions in the next screen so we can proceed with creating a Table for the Database.

# 

|  |  |
| --- | --- |
| Let us create a new **music** database…  Expand **Databases** to see the current list of databases.  Right-click **Databases** and choose **New Database…** | object explorer |

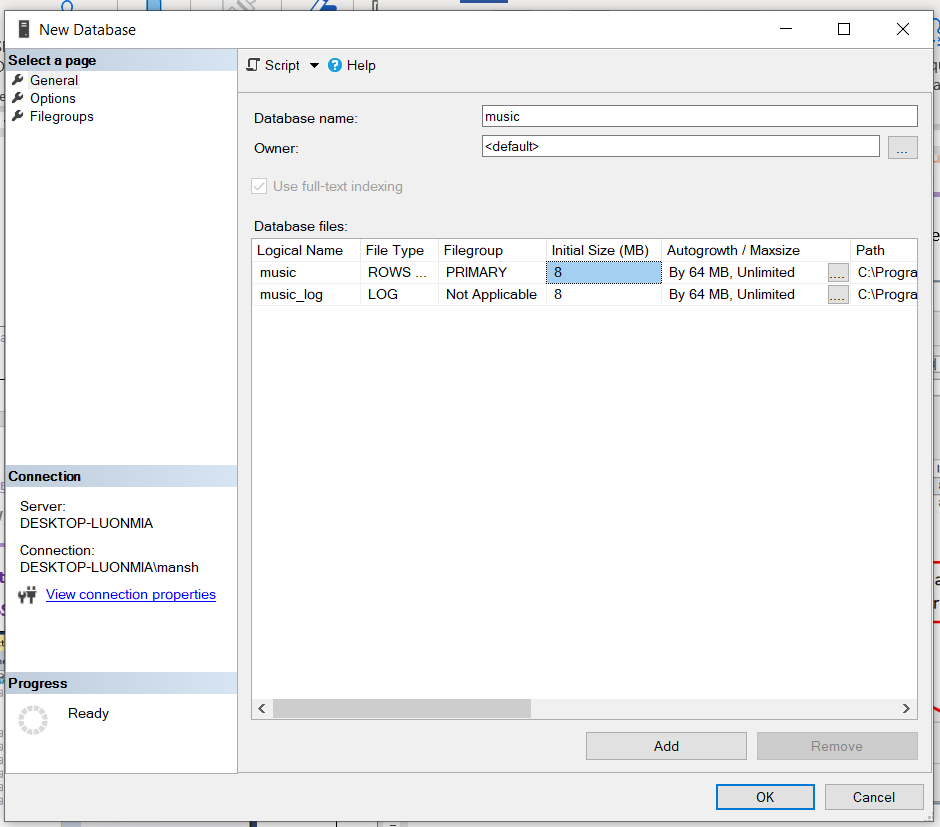
The **New Database**page opens. This is where we **name** the **database**, select the **owner** and set **options** including the MSSQL compatibility level.



You only need provide your New Database name and click OK … see 1 and 4.

It is best leaving 2 and 3. on default.

**Task:** Create a new database called **music** as shown below:



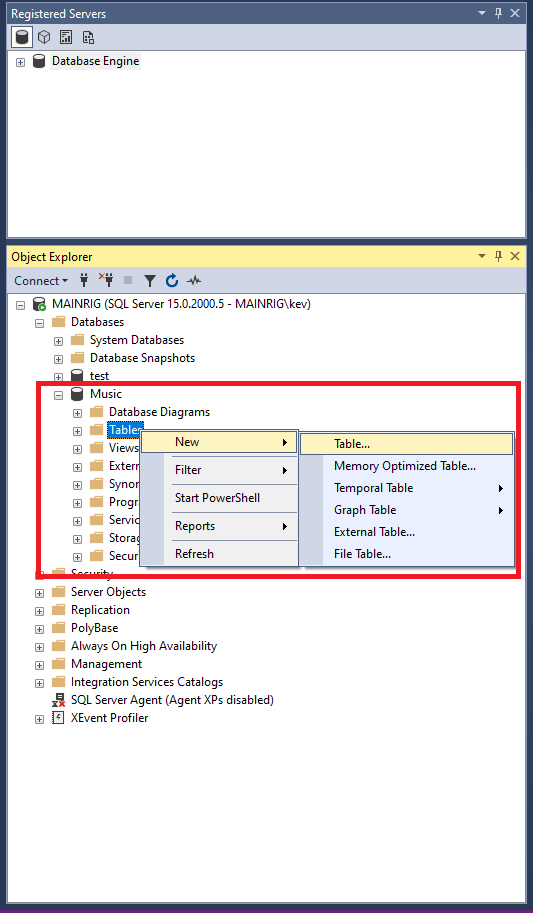
Enter the database name **music** and click **okay**

Your Database will show under the **Object Explorer** as seen below:

|  |  |
| --- | --- |
| Note the **new database** should appear in the **Object Explorer** as shown below: | You may expand the **Database** node to access any **tables, views etc** we will be creating later in the guides. |
|  | Expand to see the following in the **Object Exploer** |

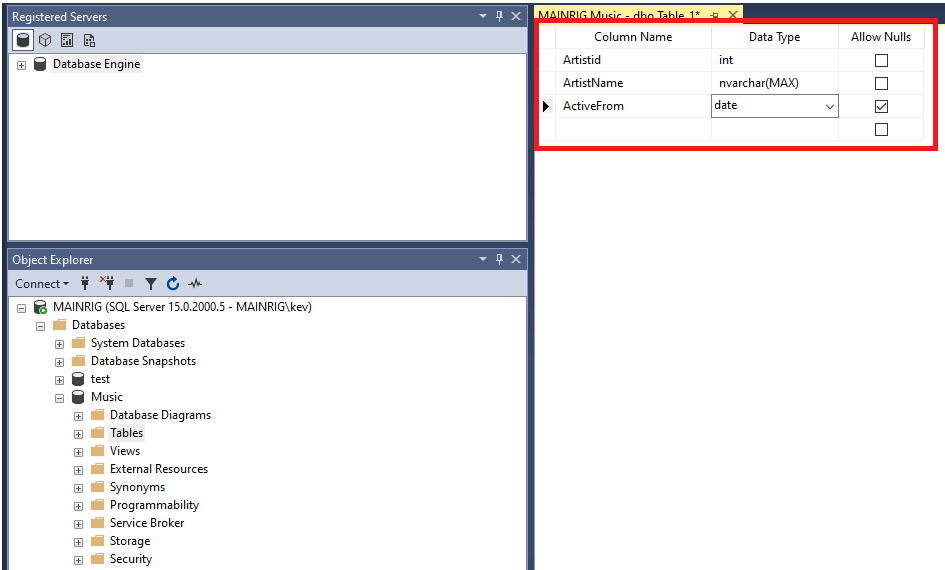
# Create a Table using the GUI – Artists table

Here, we will create a database table using the GUI method. **1. Create a New Table**



* Ensure that the correct database is expanded in the Object.
* Explorer. In our example, we'll expand the **Music** database.
* Right-click on **Tables** and select **New > Table**... from the contextual menu.

# Add Columns.

****

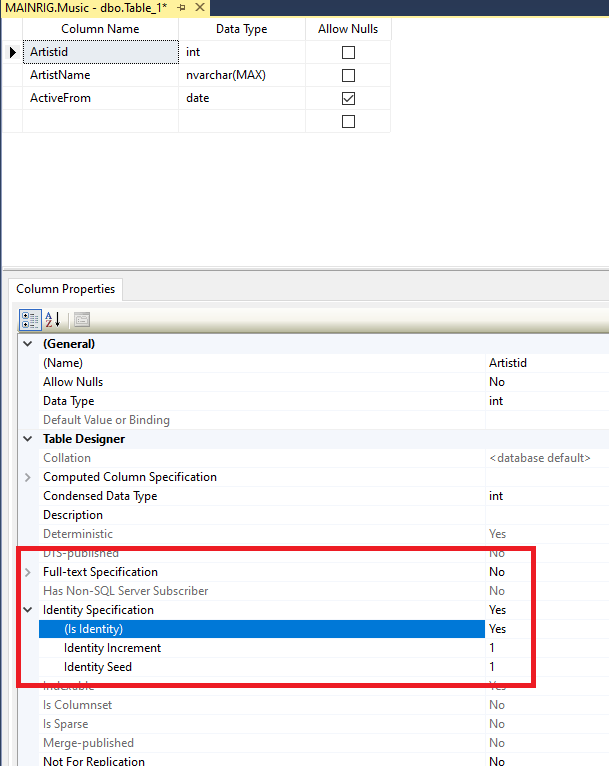
* Add the columns that need to go into the table.
* Also specify the data type for each column, as well as any other properties that you need to.

For this tutorial, we will add the following columns/data types:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Allow Nulls** |
| ArtistId | int | No |
| ArtistName | nvarchar(MAX) | No |

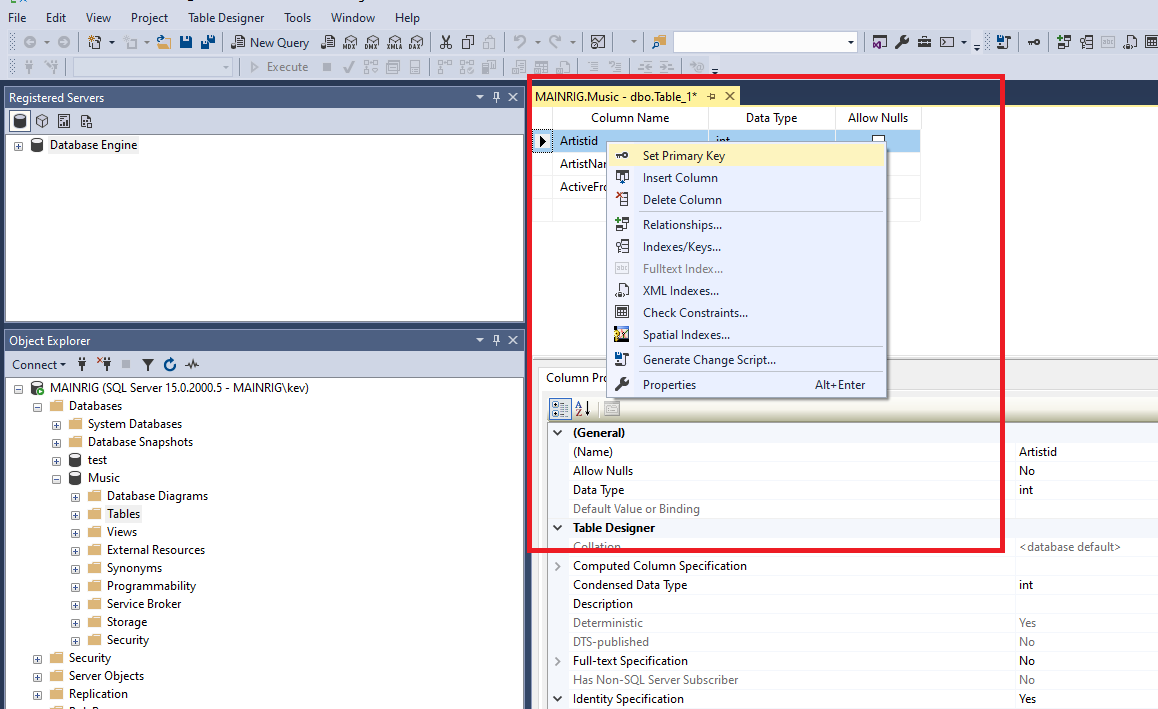
ActiveFrom Date Yes

# Set an Identity Column



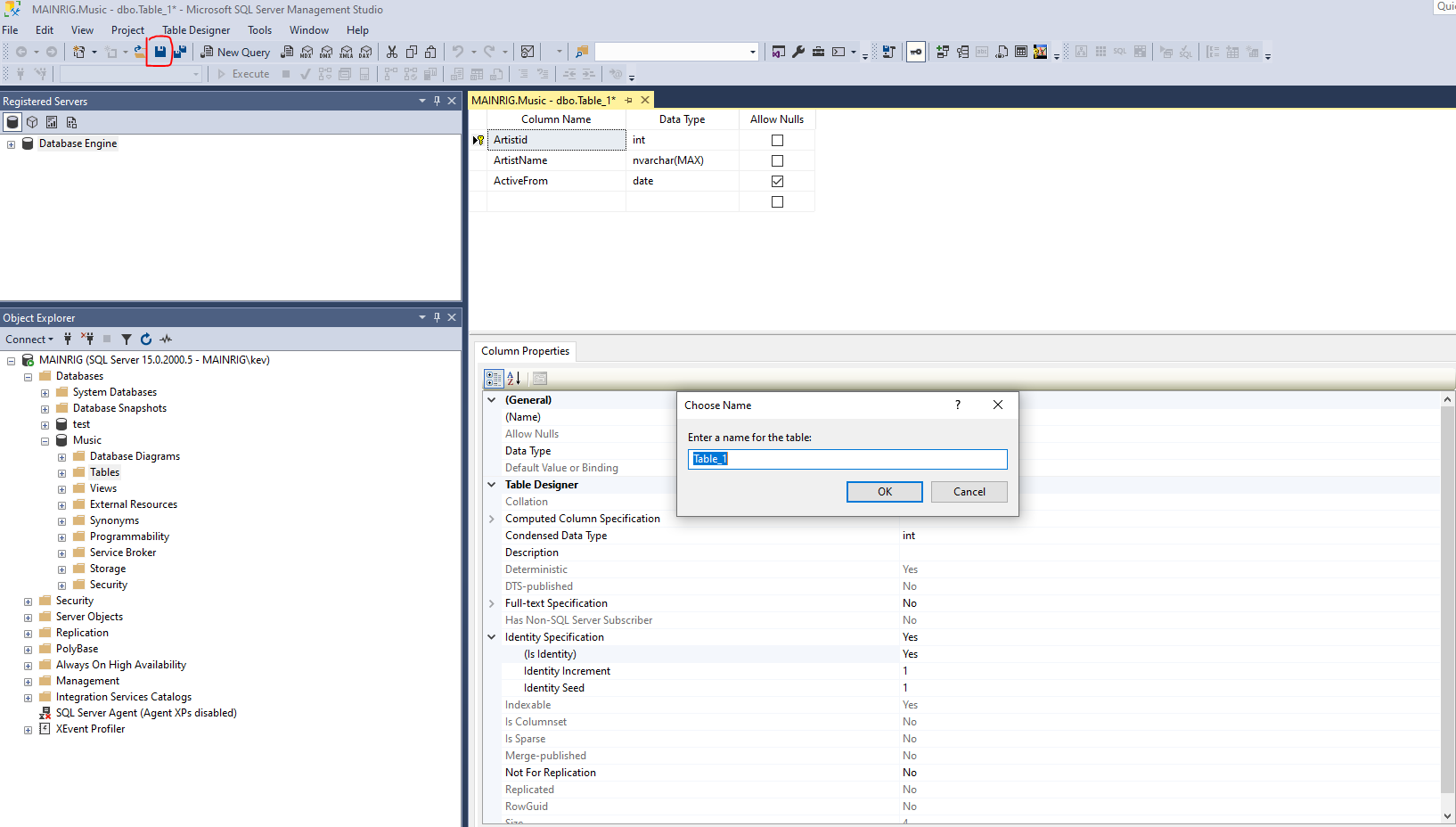
* We will now set the **ArtistId** column to be an identity column. This will make it an auto-incrementing value, so that the user does not have to insert this value.
* In the top pane, click somewhere in the **ArtistId** field.
* Now, in the bottom pane, scroll down until you see Identity Specification. Expand it, and change **(Is Identity)** to **Yes** using the drop-down list.
* Leave the other identity properties as they are.
* You can increase the height of the bottom Properties pane by clicking and dragging the top edge upwards.

# Set a Primary Key



* We will also set the **ArtistId** column to be the primary key for this table. This will be important for when we set up a relationship between this table and another.
* In the top pane, right-click in the **ArtistId** field and **select Primary Key** from the contextual menu. Once done, you will see a little key icon next to the **ArtistId** column.

# Save the Table.



* Save the table by clicking on the **Save** icon in the toolbar.
* For this tutorial, we will call the table **Artists**.
* Alternatively, you can save the table by using the File menu, pressing Ctrl+S on your keyboard, or simply clicking on the X to close the table's tab.
* All of these will prompt you to save the table.

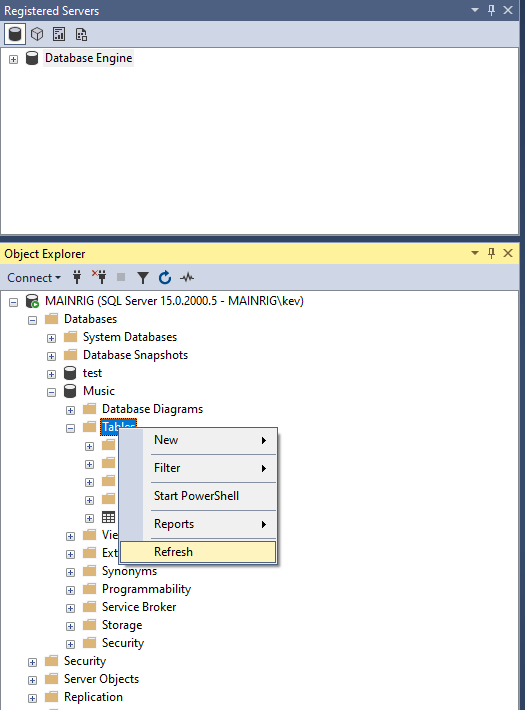
# Check that the Table was Created.

When you first open the **Tables** node in the Object Browser, your initial response might be that the table was not even created. Your table is nowhere to be seen!

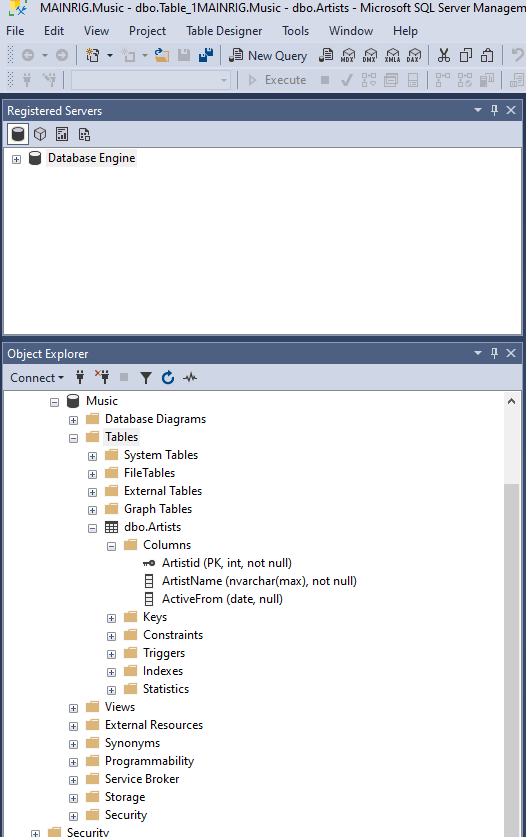
But relax... all you need to do is refresh the Object Browser.

* Right-click on the Tables node and select **Refresh** from the contextual menu.

# Refresh the Tables Node



* Ensure that the correct database is open.
* Your New Table should appear on the list after you have **refreshed**



# Create a Table using the SQL script code – genre table

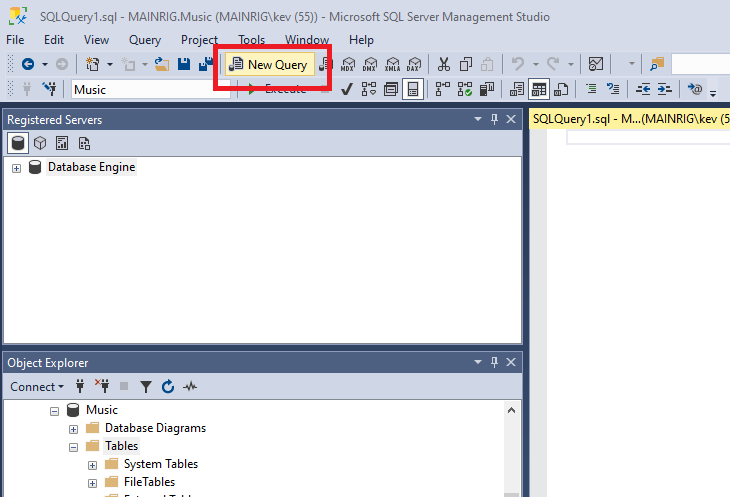
You have now created a table using the SSMS – GUI for artists. You may proceed to create the Genre and Albums.

However, with SSMS there are a number of alternative ways to create tables. In thus section we are going to see how we can generate tables using the SQL Script Code (also referred to as Transactional-SQL – TSQL and is part of SQL Data Definition Language set.

Having just [created a table via the SSMS GUI,](http://www.quackit.com/sql_server/sql_server_2016/tutorial/create_a_table_in_sql_server_2016.cfm) we will now create a table (or two) using an SQL script.

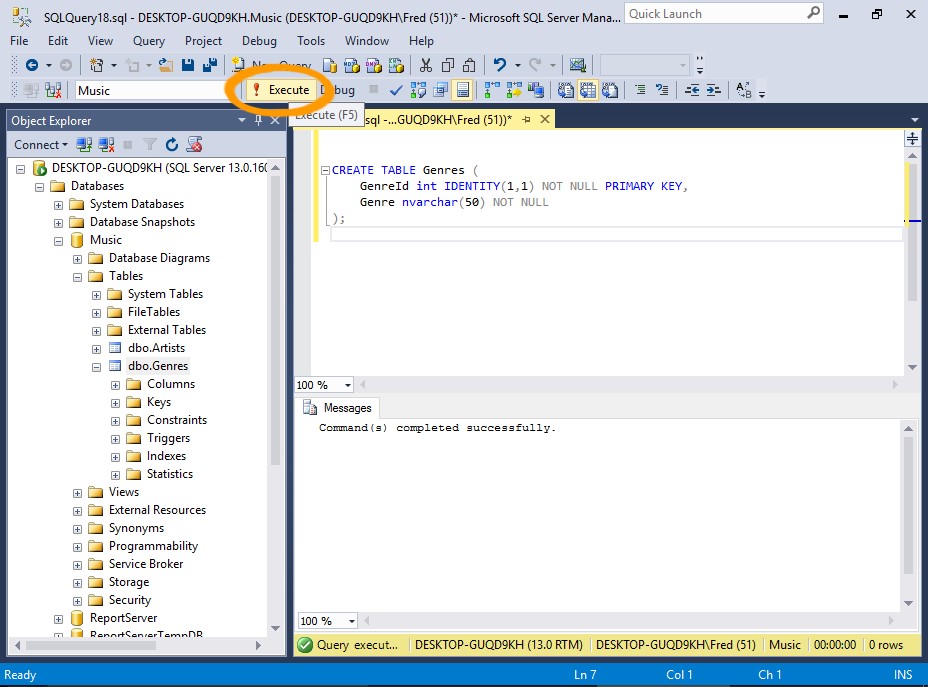
To create a table using SQL, use the CREATE TABLE statement. Provide the column names and their data types, plus any other properties as required.

# Open a New Query Window



* Open a new query window by clicking on New Query the toolbar.

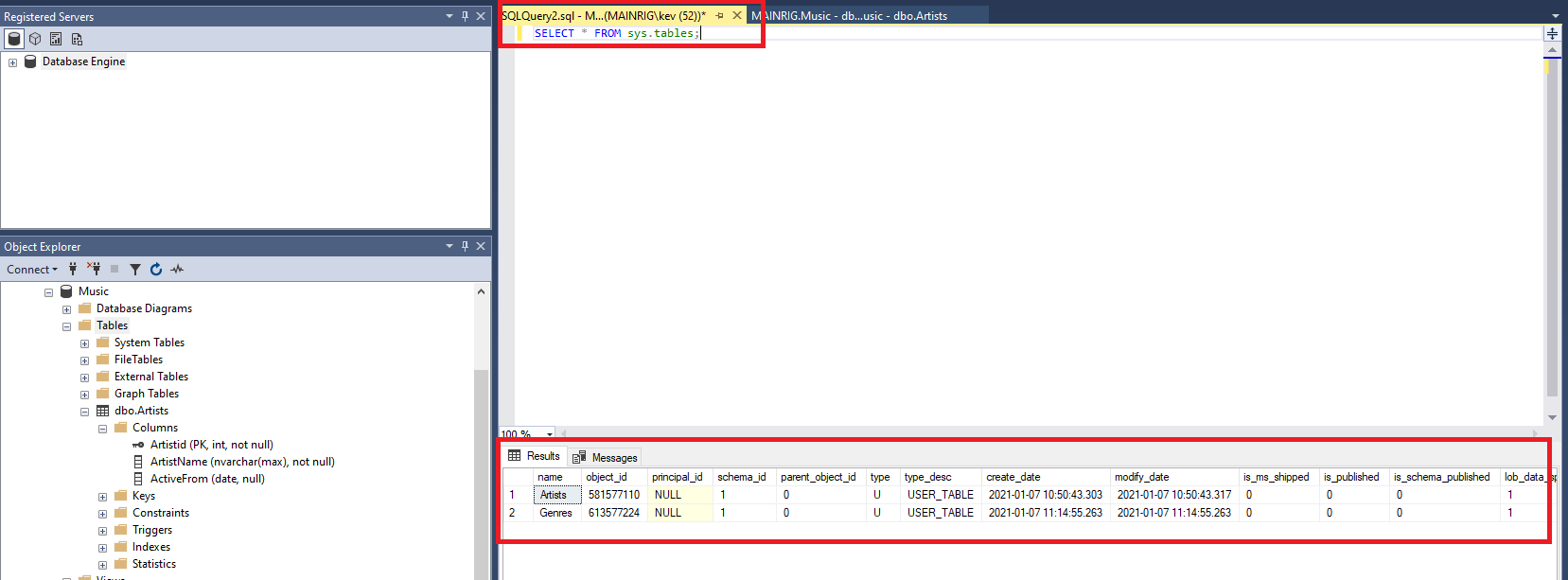
# Run the SQL Script



* Run the SQL script (below) by pasting it into the query window, then clicking the Execute button on the toolbar. You can also run a query by:
* Pressing F5 on your keyboard.
* Clicking Query > Execute from the top menu.
* Right clicking in the actual query window and selecting Execute from the contextual menu.

|  |
| --- |
| -- Task: (1) **Click** the **New Query** tab:  -- (2) **Copy** n **Paste** the below **TSQL** code into a new **Query Window.**  -- (3) Click on E**xecute** to run the **TSQL**: |
| CREATE TABLE Genres (  GenreId int IDENTITY(1,1) NOT NULL PRIMARY KEY,  Genre nvarchar(50) NOT NULL  ); |

# Check that the Table was Created.



* Let us also use SQL to check that the table was created. refresh the Object Explorer by right clicking on the Tables node and selecting Refresh like we did when we created our previous Table.

# Table Options

The CREATE TABLE statement accepts many additional options that allow you to specify the exact properties of the table.

There are too many to go into detail in this tutorial, but they include options for encryption, replication, indexes, and more.

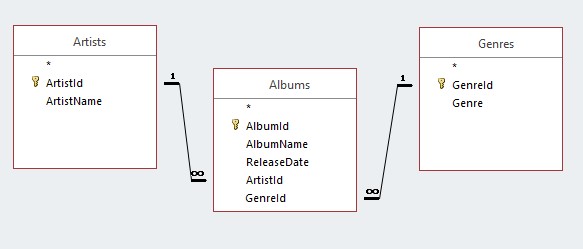
However, we will be covering one of these options next — foreign key constraints — when we **create a relationship**between tables later in the guide.

--------------------------------------------------------------------------------------------------------------------------

# Create a relationship between tables by using the GUI or SQL script.

In relational database design, a ***relationship*** is where **two or more tables** are linked together because they contain related data. This enables users to run queries for related data across multiple tables.

Here, we will create the following relationships.



# The Method

Here's how we'll do it:

* We will use **SQL** to create the Albums table and one relationship.
* We will use the **GUI** to create the other relationship.

That way, you'll get to see both methods of creating a relationship.

We only need to create one table because we've already created two of these tables previously in this tutorial (the Artists table **via the GUI**and the Genres table **using SQL**[)](http://www.quackit.com/sql_server/sql_server_2016/tutorial/create_a_table_from_an_sql_script_in_sql_server_2016.cfm).

# Create a Relationship using SQL

Open a new query window in SSMS and run the following code:

|  |
| --- |
| -- Task: (1) **Click** the **New Query** tab:  -- (2) **Copy** n **Paste** the below **TSQL** code into a new **Query Window.**  -- (3) Click on E**xecute** to run the **TSQL**: |
| -- The first part of that statement creates the table.  CREATE TABLE Albums  (  AlbumId int IDENTITY(1,1) NOT NULL PRIMARY KEY,  AlbumName nvarchar(255) NOT NULL,  ReleaseDate date NOT NULL,  ArtistId int NOT NULL, GenreId int NOT NULL  -- The last part defines the relationship. This part:  CONSTRAINT FK\_Albums\_Artists FOREIGN KEY (ArtistId)  REFERENCES dbo.Artists (ArtistId)  ON DELETE NO ACTION  ON UPDATE NO ACTION  ); |

# Some explanation on the SQL Script Code.

Do not worry as the intent is to also have you familiar with the syntax of TSQL the scripting code or SQL programming language. SSMS is intuitive and we will explore how to auto generate SQL code later in the guides.

The first two lines create the relationship. They create a foreign key constraint between the Albums.ArtistId column and the Artists.ArtistId column.

The last two lines specify what SQL Server should do if someone tries to delete or update a parent record that is being referenced by a record in the child table. In this case, NO ACTION means that the delete/update won't go ahead. The user will just get an error.

You could change this to ON DELETE CASCADE if you want to be able to delete the parent and the child in one go (i.e. the delete will cascade from the parent to the child). The same logic applies to updates, by using ON UPDATE CASADE.

NO ACTION is the default value, so we could've done without those last two lines of code. However, I included it, because it's an important factor to think about when creating foreign key constraints.

# What's a Foreign Key Constraint?

Some background on relationship and constraints:

A *foreign key constraint* defines a relationship between this table and another table. When you create a foreign key constraint, you create it against a specific column in the *child* table, to reference a specific column in *parent* table.

This makes the column in the child table a *foreign key*. The constraint ensures that any value that goes into this (foreign key) column corresponds with a value in the primary key column of the parent table. If someone tries to enter a value that doesn't correspond with a value in the parent table's primary key column, SQL Server will throw an error.

This helps enforce referential integrity. It prevents us from having orphaned records (child records that have no parent). Or in our example, albums that aren't associated with any artist.

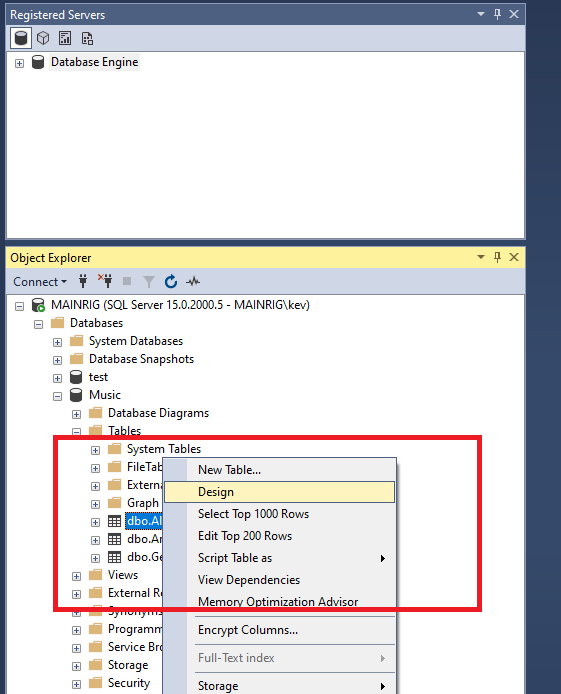
|  |
| --- |
| What does this mean in practice?  You need to ensure primary key data exists before posting the value as a foreign in a related table. |

# Create a Relationship via the GUI

Now we'll create the other relationship via the SQL Server Management Studio SSMS - GUI.

It would have been easier to include this in the above script, but the purpose is to demonstrate both methods of creating a relationship.

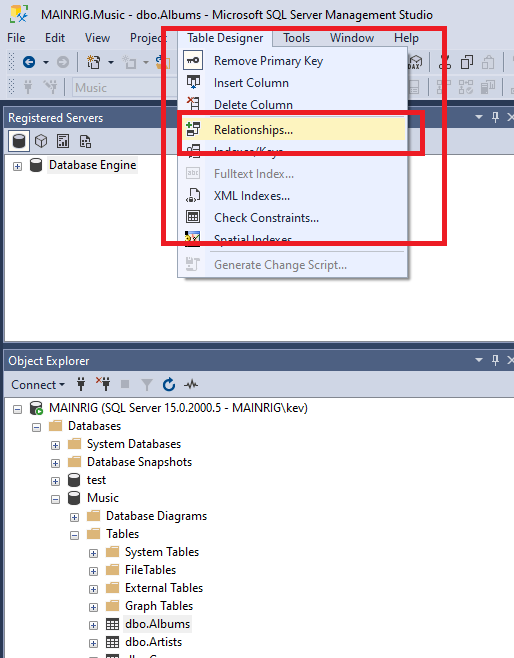
# Open the Child Table in the Table Designer



* Right-click on the child table (our newly created Albums table) and select Design from the contextual menu.

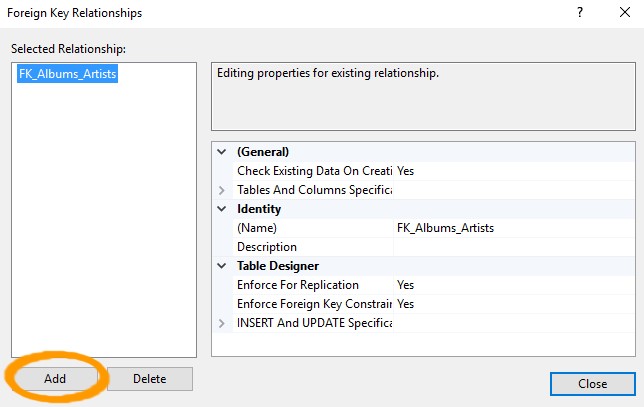
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| If you can't see your newly created table in the Object Browser, you probably need to refresh the Object Browser. | | | | |
| Right-click on the | Tables | node and select | Refresh | . |

# Open the Foreign Key Relationships Dialog



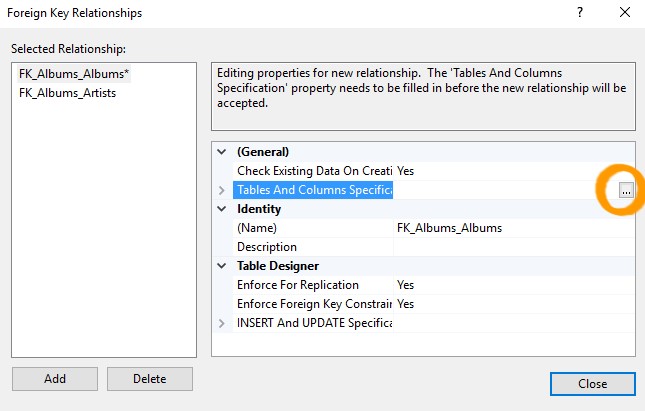
* Select Table Designer > Relationships... from the top menu.

# Add the Relationship



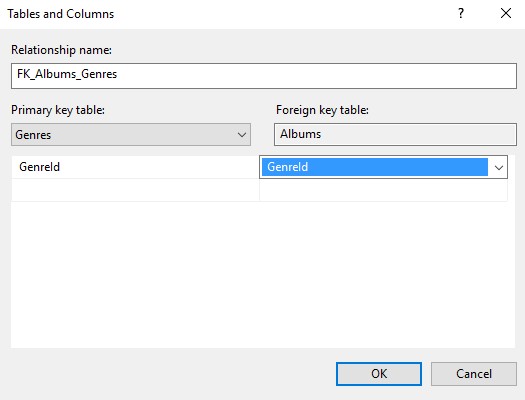
* The Foreign Key Relationships dialog will show you any existing relationships for the table.
* We can see the relationship that we established just before when we created the table. Click Add to add another relationship.

# Select Tables And Columns Specification



* A new relationship appears above the other one in the Selected Relationship list with a name of FK\_Albums\_Albums.
* Ensuring that the the new relationship is selected, click Tables And Columns Specification in the right pane.
* An ellipse appears to the right of the property.
* Click the ellipses (...) to launch the Tables and Columns dialog box.

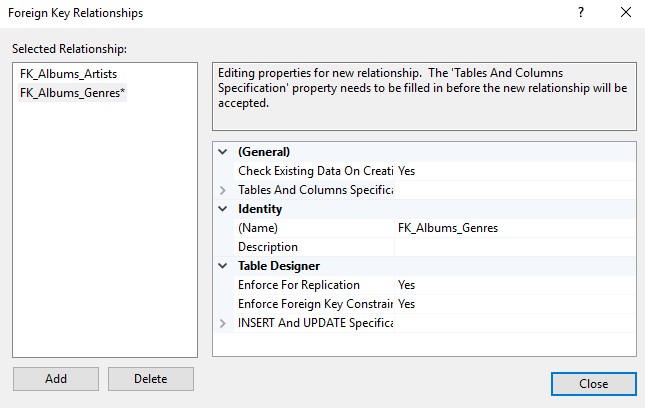
# The Tables and Columns Dialog Box



* Here, you select the primary key table on the left pane, and the foreign key table on the right
* Under Primary key table: select Genres as the table and GenreId as the column.
* Under Foreign key table: select Albums as the table and GenreId as the column.
* Click OK.

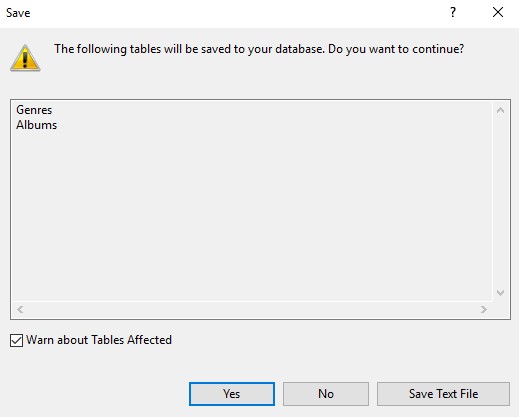
SQL Server will suggest a name for the relationship. You can edit this if you wish. Otherwise, leave it as it is.

# The Relationship



* Your relationship will now be displayed correctly in the Foreign Key Relationships dialog box. Click Close.

# Saving The Relationship



Your relationship won't be saved until you save the table.

When you save the table, you will probably get a warning that two tables will be saved. This is to be expected, as the relationship affects two tables.

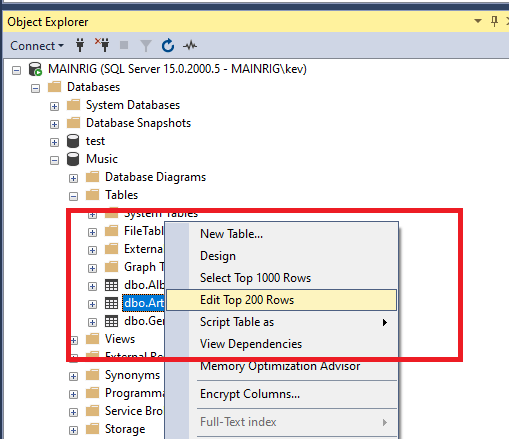
If you select Table Designer > Relationships… for the parent table, you will also see the relationship there.

# Insert data using a number of methods such as directly, copy/paste, import, use SQL, or use an external application.

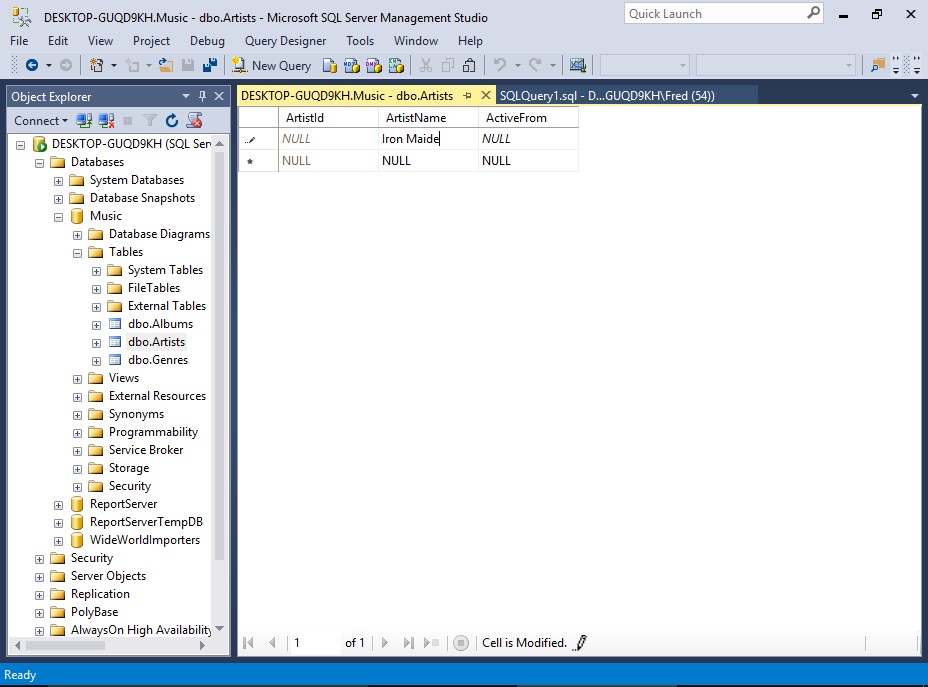
Here, we'll take a quick look at the various methods of inserting data into SQL Server.

From a development view point we need to incorporate test data.

# Direct Input



* Using this method, you enter text directly into the table using the SSMS GUI.
* First, using the SSMS GUI, navigate to the table you want to enter data into.
* Right-click on the table and select Edit Top 200 Rows.



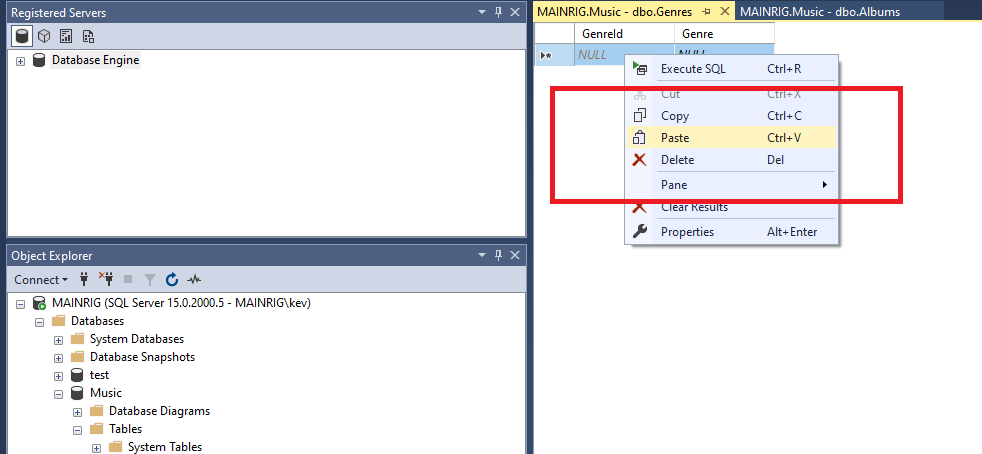
The table will open, allowing you to type data directly into the cells.

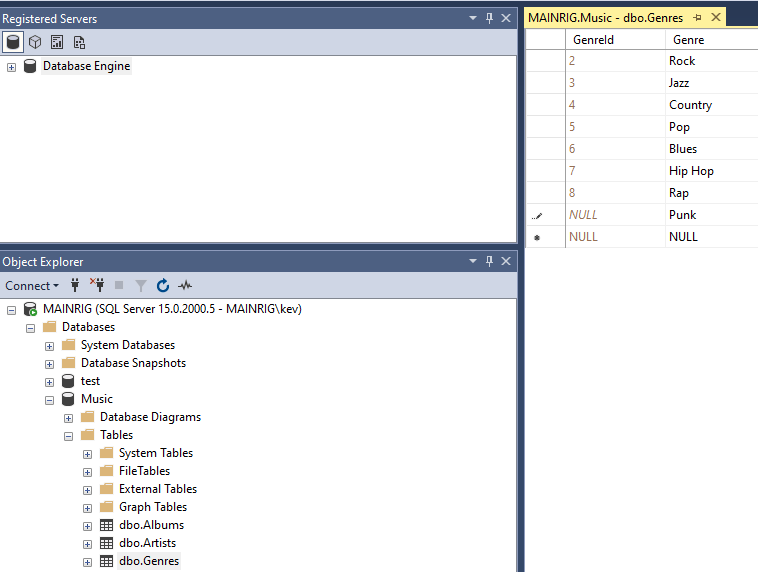
You won't need to enter data into columns with auto-generated values, such as identity columns, timestamp columns, etc.

Actually, SQL Server will usually prevent you from entering data into such columns.

|  |
| --- |
| * Copy/Paste * First, copy the data from the external source, which is provided below: * We can then paste the following data into the **Genres table:** |
| 1. Rock 2. Jazz 3. Country 4. Pop 5. Blues 6. Hip Hop 7. Rap 8. Punk |

* Now, switch to SSMS and open the Genre table by using the previous method (i.e. right-click on the table and select Edit Top 200 Rows).
* Once the table has opened, right-click in the document window and select Paste. You will have to do this for each item.





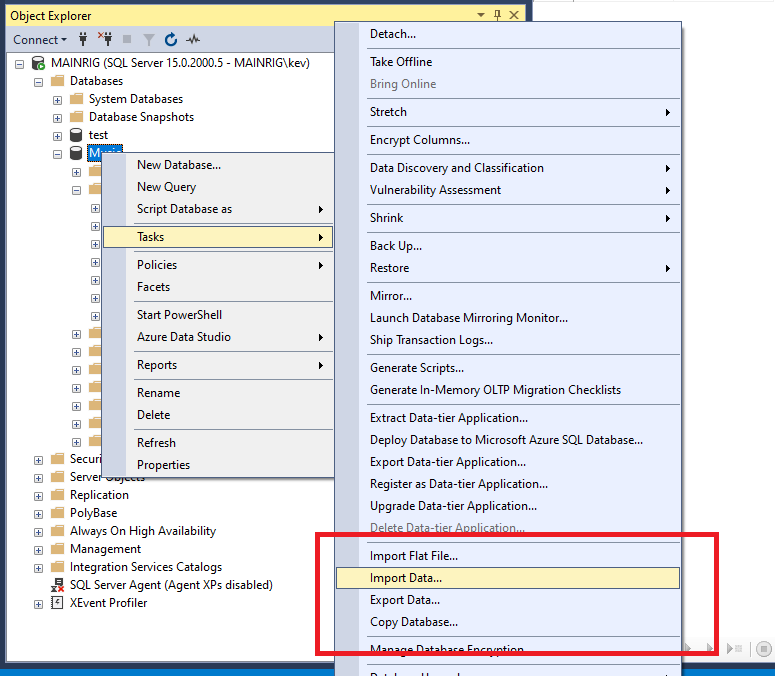
SQL Server will ignore the first column, because it's an identity column. So if you already have records in that table, the resulting IDs for the new data will be different to the above.

# Import Data

SQL Server Management Studio includes an import wizard that helps you import data into a database.

We'll be covering this method **next**[**,**](http://www.quackit.com/sql_server/sql_server_2016/tutorial/import_data_in_sql_server_2016.cfm) but here's a hint of what's to come.

* Right-click on the database and select Tasks > Import Data...



More details and screenshots about Import Data in the coming Quick Guide documents.

# Use SQL to Insert the Data

You can use the **SQL INSERT**statement to insert data into a table.

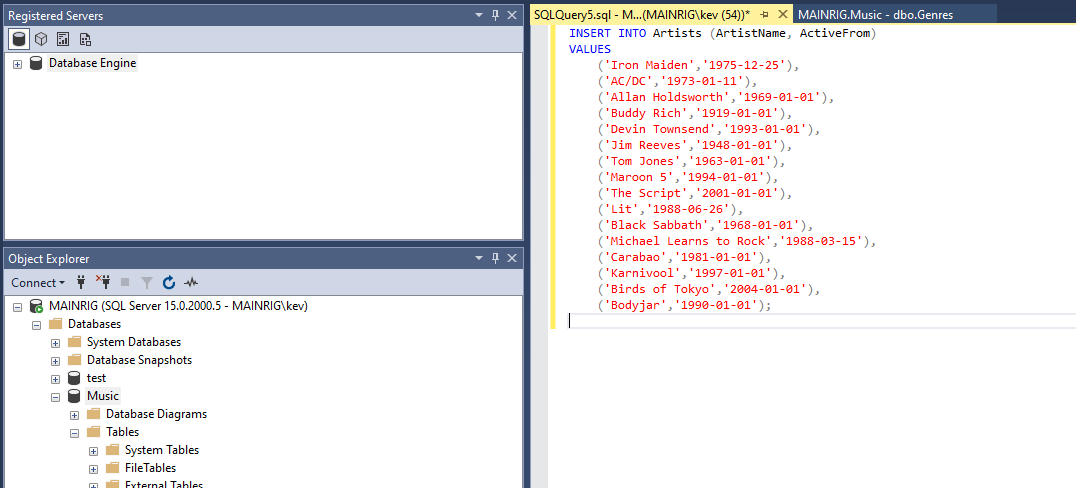
To do this, open a new query window, type the SQL, then execute the statement (sample statement below).

In our case, the first column is an identity column, so we won't insert any data for that column. Therefore, we need to specify the actual columns that we want to insert the data into (i.e. the table has three columns but we're only inserting two).

To insert multiple rows, there are a few ways to go about this. One way is to add multiple SQL INSERT statements — one for each row to be inserted.

Another way is to separate each set of values with a comma. Like our sample SQL statement below. Sample SQL Statement

|  |
| --- |
| -- Task: (1) **Click** the **New Query** tab:  -- (2) **Copy** n **Paste** the below **TSQL** code into a new **Query Window.**  -- (3) Click on E**xecute** to run the **TSQL**: |
| -- SQL Insert command for data entry  INSERT INTO Artists (ArtistName, ActiveFrom)  VALUES  ('Iron Maiden','1975-12-25'),  ('AC/DC','1973-01-11'),  ('Allan Holdsworth','1969-01-01'),  ('Buddy Rich','1919-01-01'),  ('Devin Townsend','1993-01-01'),  ('Jim Reeves','1948-01-01'),  ('Tom Jones','1963-01-01'),  ('Maroon 5','1994-01-01'),  ('The Script','2001-01-01'),  ('Lit','1988-06-26'),  ('Black Sabbath','1968-01-01'),  ('Michael Learns to Rock','1988-03-15'),  ('Carabao','1981-01-01'),  ('Karnivool','1997-01-01'),  ('Birds of Tokyo','2004-01-01'),  ('Bodyjar','1990-01-01'); |



# Website/Application

**Reference Only:**

One of the most common methods of adding data to a client/server database like SQL Server is via a Web and Mobile Apps.

Note it is important to remember SQL Server is a backend Database repository.

Most of the worlds Development platforms can connect into SQL Server to access Data.

The application could be anything, from a corporate CRM to a customer facing website. Data is added to SQL Server via the application, which generates the code to insert the data into the database.

This method is similar to the above SQL method, because most applications generate SQL code in order to insert data into the database. The same applies when reading from the database.

# SQL Server Management Studio import wizard to help you import data from an external source.

You can import data to your SQL Server database from various other sources, including Excel spreadsheets, CSV files, and other databases such as Oracle, Access, and other SQL Server instances.

# 1. Import a CSV File.

Right now, we'll import a CSV file into our Albums table that we created in the earlier Quick Guide when we created a relationship.

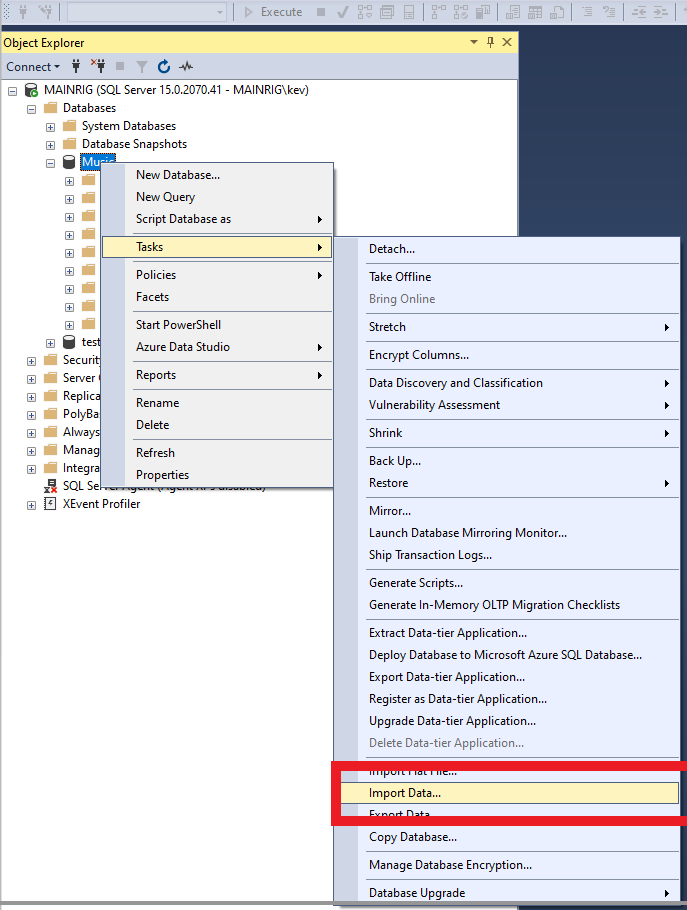
Included in the quick pack there is a Excel file called: **Albums**

The file contains the following contents:

|  |
| --- |
| AlbumId,AlbumName,ReleaseDate,ArtistId,GenreId  1,Powerslave,1984-09-03,1,1  2,Powerage,1978-05-05,2,1  3,Singing Down the Lane,1956-01-01,6,3  4,Ziltoid the Omniscient,2007-05-21,5,1  5,Casualties of Cool,2014-05-14,5,1  6,Epicloud,2012-09-18,5,1  7,Somewhere in Time,1986-09-29,1,1  8,Piece of Mind,1983-05-16,1,1  9,Killers,1981-02-02,1,1  10,No Prayer for the Dying,1990-10-01,1,1  11,No Sound Without Silence,2014-09-12,9,4  12,Big Swing Face,1967-06-01,4,2  13,Blue Night,2000-11-01,12,4  14,Eternity,2008-10-27,12,4  15,Scandinavia,2012-06-11,12,4  16,Long Lost Suitcase,2015-10-09,7,4  17,Praise & Blame,2010-06-26,7,4  18,Along Came Jones,1965-05-21,7,4  19,All Night Wrong,2002-05-05,3,2  20,The Sixteen Men of Tain,2000-03-20,3,2 |

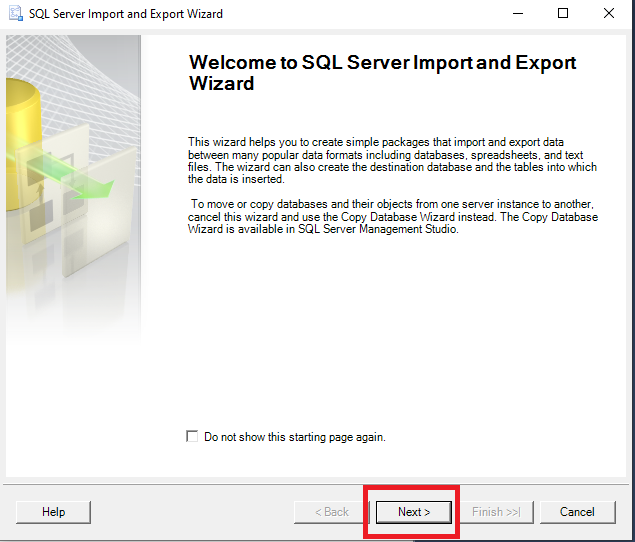
Either download the Albums CSV file or copy and paste the above into Excel and save as a CSV file and save the above file into a location that you will be able to get to from the SQL Server Import and Export Wizard, and let's get started.

# 1. Import Data.



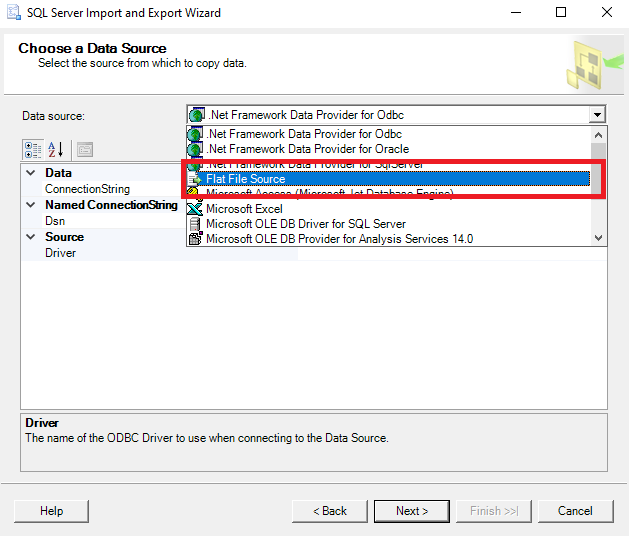
* Right-click on the database and select Tasks > Import Data...
* You can also access the import wizard from various other places, including the Windows Start menu, the command prompt, and Visual Studio with SQL Server Data Tools (SSDT).

# Welcome to the SQL Server Import and Export Wizard.

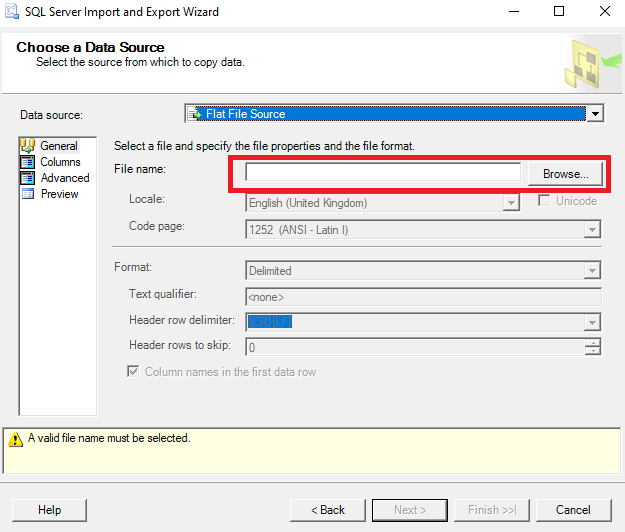


* The SQL Server Import and Export Wizard start page is displayed.
* Click Next >.
* Check “Do not show this starting page again” if you want to save time and clicks in the future.

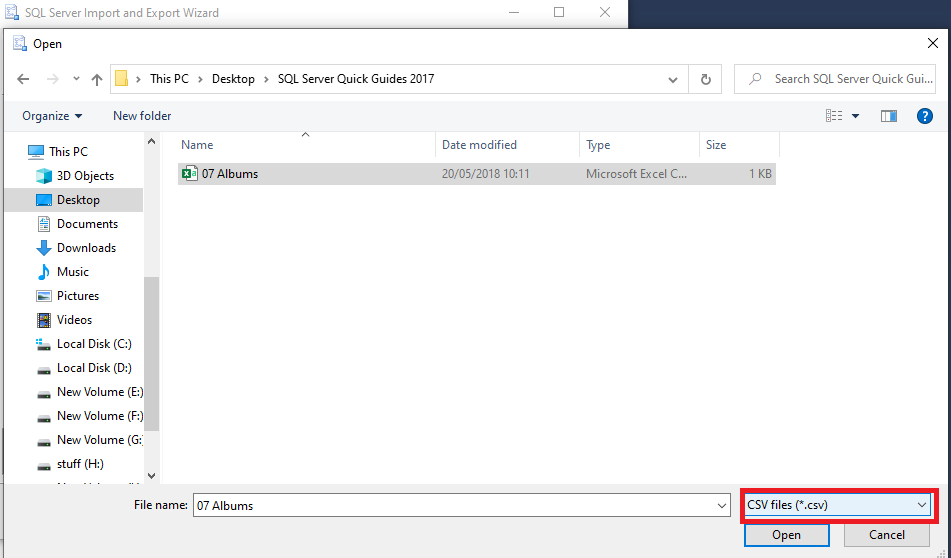
# Choose a Data Source.

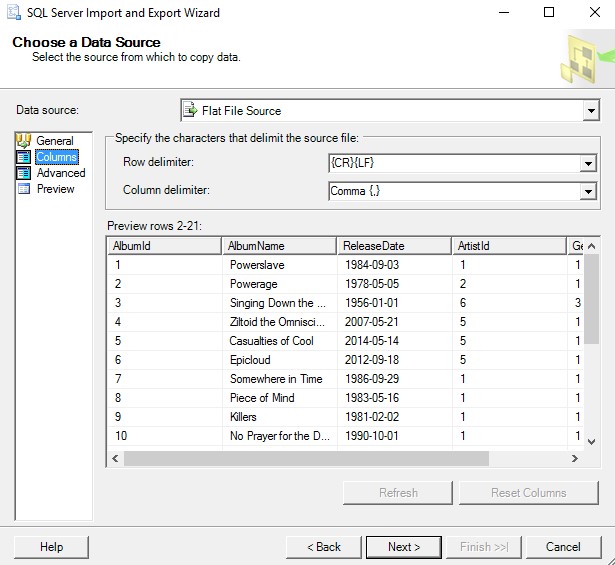


* Then Browse for your **album.csv** file as shown below:

****

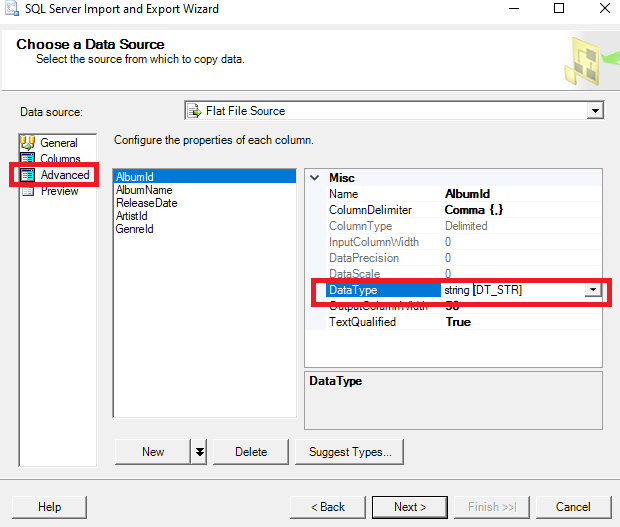
* Make sure that the file type is correct as shown below

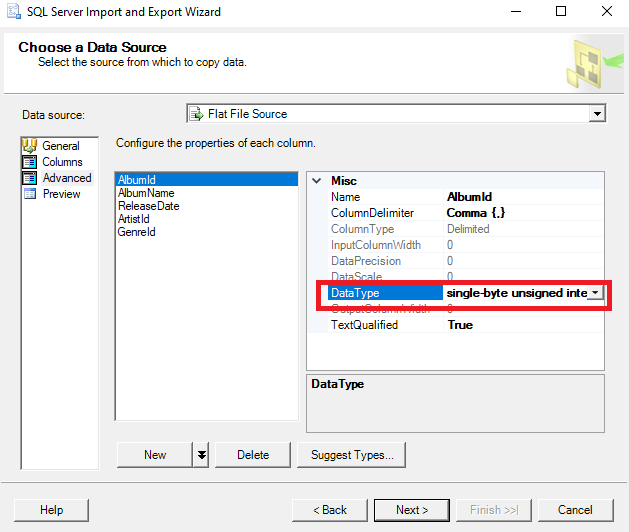
****



* Select the **file type** and its **location**.
* Ours is a CSV file, so select **Flat File Source** then browse to the file's location.
* Also click on **column** in the left menu to check the **delimiter** settings, and the other options too if you like. The default settings should be fine in our case.

# Advanced Settings.





* Check the advanced settings (by clicking on **Advanced** in the left menu) to make sure nothing looks out of the ordinary.
* Click on each column name to view the properties for that column. In our case, you may find that all columns are listed with a data type of string [DT\_STR], which could cause problems later on in the wizard.
* Take this opportunity to change the data types to the following:

AlbumId single-byte unsigned integer [DT\_UI1]

AlbumName Unicode string [DT\_WSTR]

ReleaseDate database date [DT\_DBDATE]

ArtistId single-byte unsigned integer [DT\_UI1]

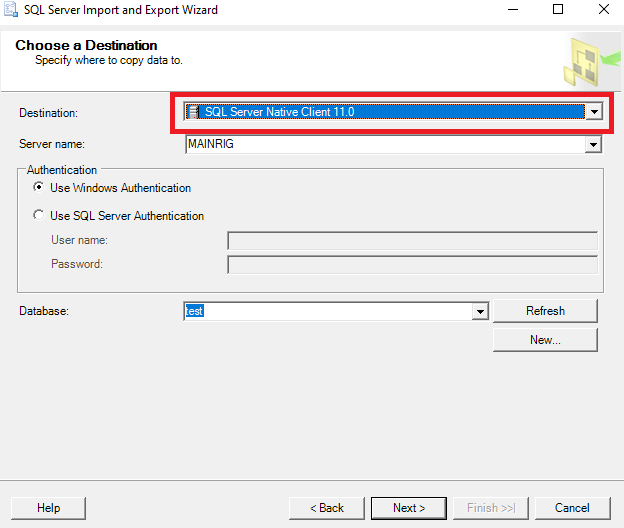
GenreId single-byte unsigned integer [DT\_UI1]

You can also use the suggest types… However, you may still need to modify some columns manually.

The wizard will warn you later on if it thinks there will be any issues converting data types from the source file. If that happens, you may need to come back and modify your selection here.

When you have made the changes above click next

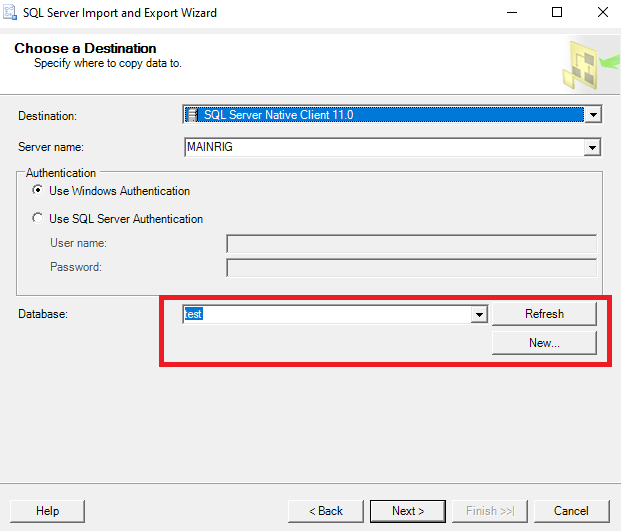
# Choose a Destination.



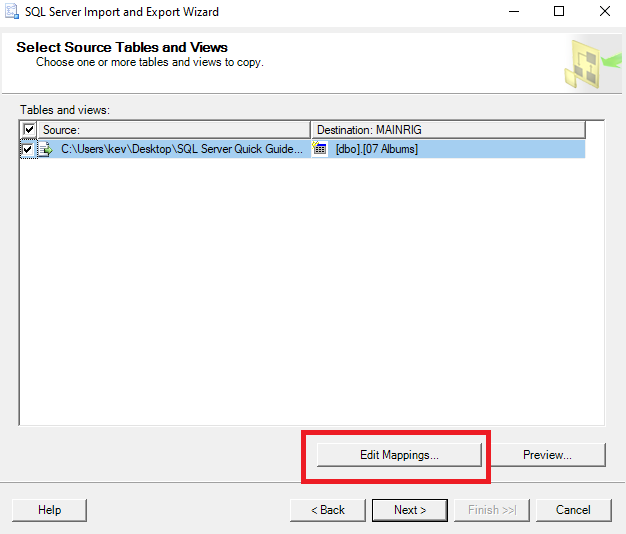
* Here, you choose the data provider that matches the data storage format of the destination.
* There are various options for SQL Server (SQL Server Native

Client, the Microsoft OLE DB Provider for SQL Server, or the .NET Framework Data Provider for SQL Server). We will choose the native client.

* Select the SQL Server Native Client option drop-down list. If you need to change the Authentication mode and/or the destination database, do so now.

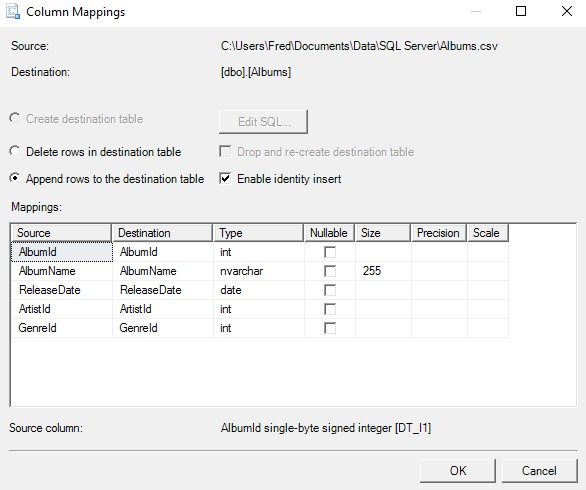


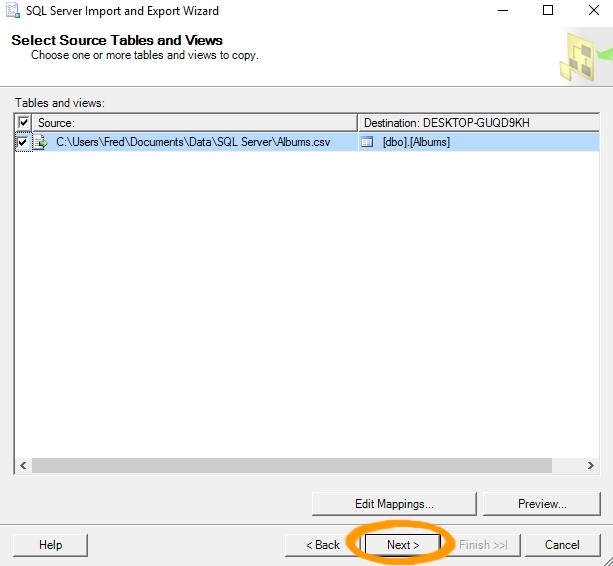
* Select your Database as above.
* Click Next > to continue.
* Select Source Tables and Views.



* Here, you get to select the source and destination.
* Ours is a simple one and everything is selected correctly by default. However, before continuing, click Edit Mappings... to review the column mappings.

# Column Mappings.





* Check that the column mappings and other details are correct.

For our example, we need to make sure that Append rows to destination table is checked, and that the source and destination columns match, and with the correct data type. Also check Enable identity insert to keep our existing values for the primary key column. This will insert the values from our file, rather than have SQL Server auto-generate them.

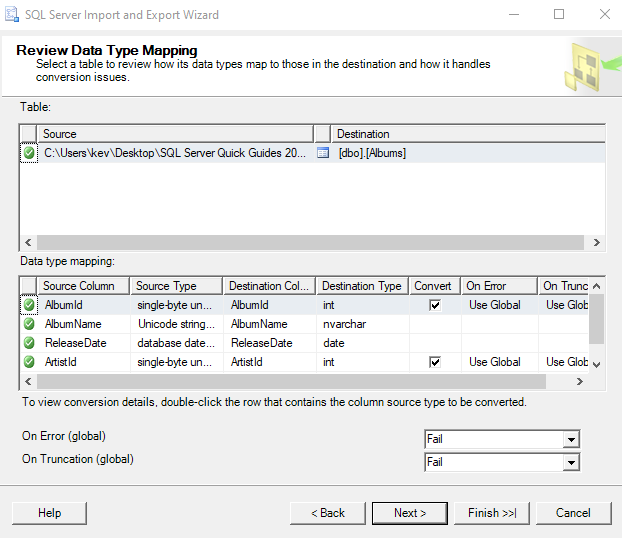
* Click OK or Cancel to close the dialog box.

Then, once back in the Select Source Tables and Views dialog box, click Next > to continue.

If Create destination table is selected and the Append rows to destination table option is greyed out, check that the name of the source file matches the name of the destination table. Hint, if you click on destination in tables and views it will list the tables available.

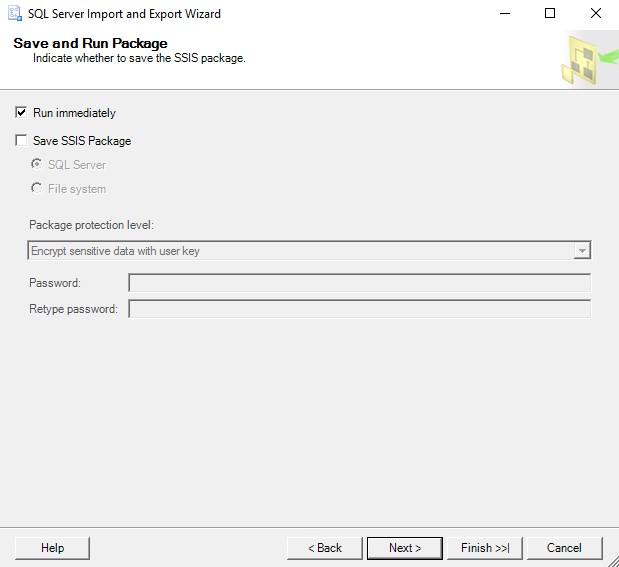
* Then click next.

# Review Data Type Mapping.



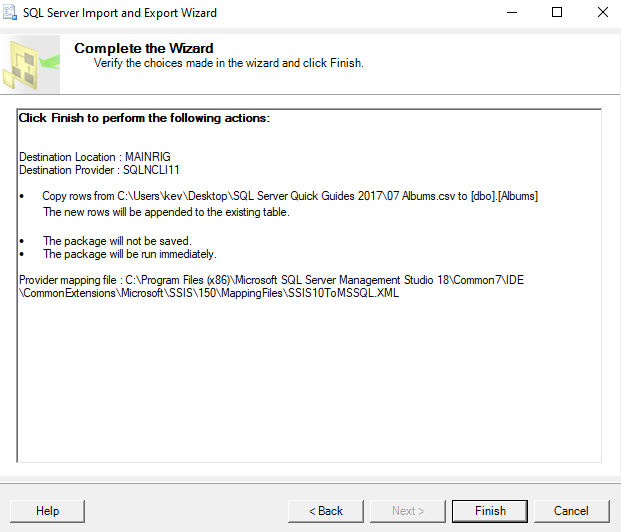
* Here is our chance to review the data type mappings that we set up earlier.
* If the wizard thinks there could be a problem with converting data types, it will warn you on this screen.
* If all looks good, click Next >.

# Save and Run the Package.



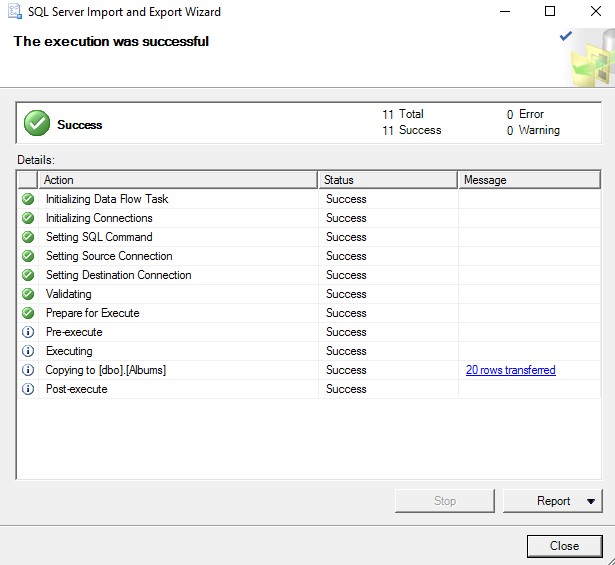
* The wizard gives you the option of saving the package so that you can use it again later.
* In our case, we will not bother saving it.
* Just keep Run immediately selected and click Next >.

# Complete the Wizard.



* One last chance to review all actions. If it all looks good, click Finish.

# Execution Report: Success or Failure.

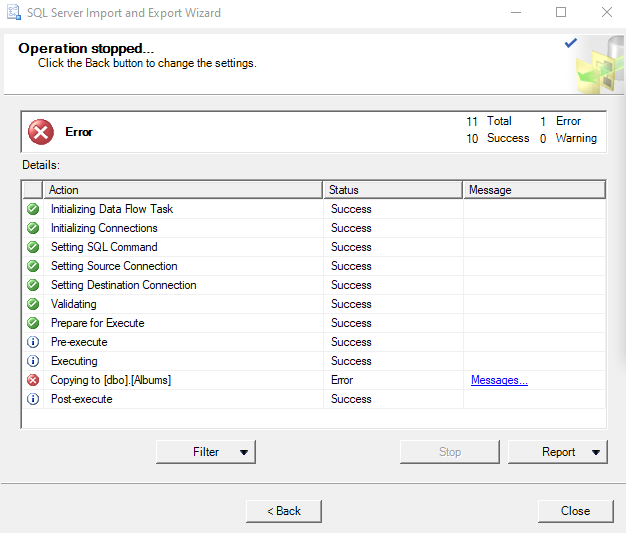


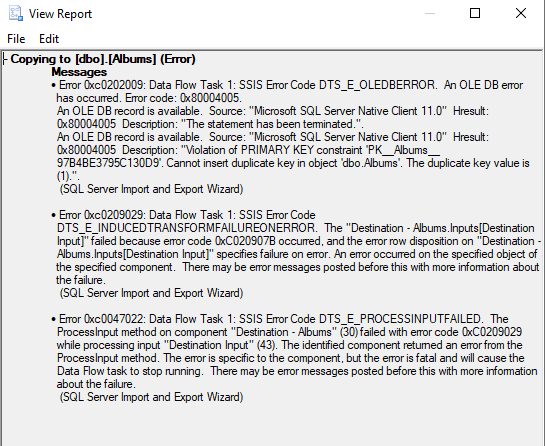
The wizard will now try to execute the import. If there were any errors, they will be listed on this screen, and you will need to go back and correct the cause, before running it again.

If you get the execution was successful, click **close**.

You can also save a report of the import by using the Report option on this screen.

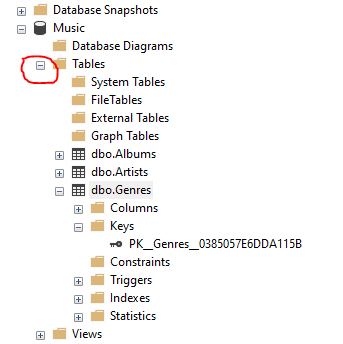
If you happen to run in to the following error. Follow the steps below.



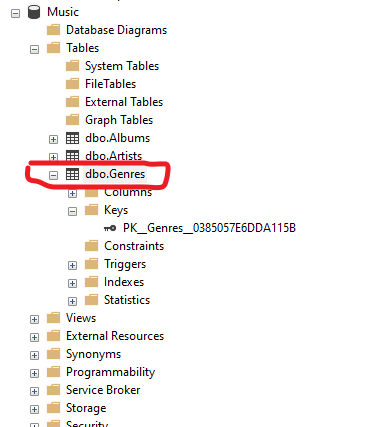


To fix the above issue do the following.

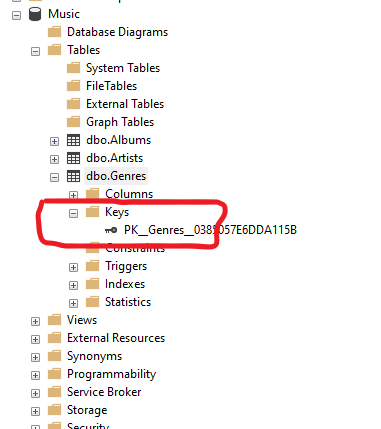
In Object Explorer, expand Tables by clicking the + symbol.



And then do the same to dbo.Genres



And then again to Keys



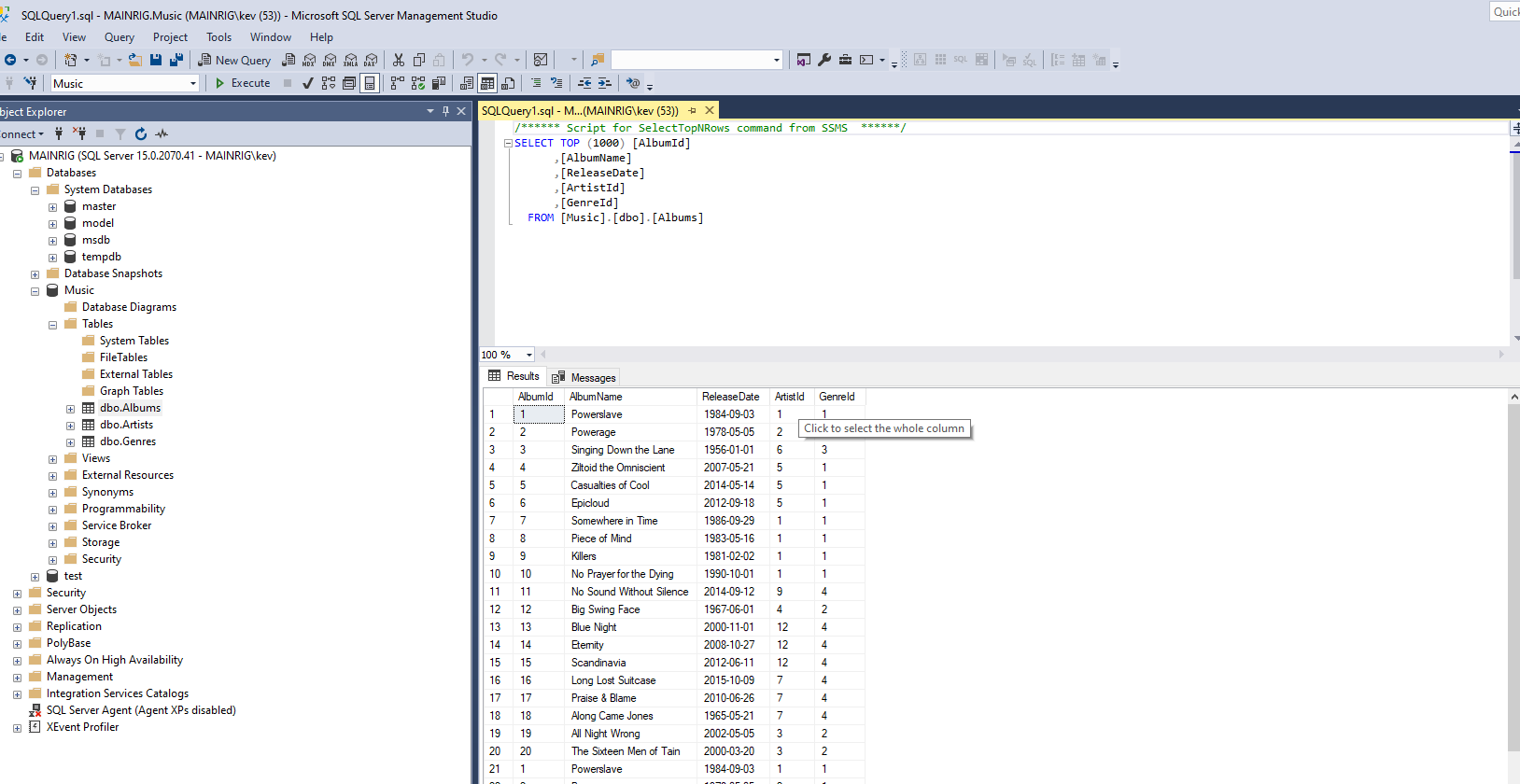
Double click PK\_Genres and remove Primary Key and any Foreign Keys if they are present.

Now try to import the data again.

Make a note of any Keys removed as you will need to reapply them afterwards.

**Don’t forget to Refresh the Database**.

# Check the Data.



Now run a quick query to see for yourself that the data has in fact been inserted.

You can either use the Select Top 1000 Rows option (after right clicking on the Table)

Alternatively you can do this via **New Query**

|  |
| --- |
| -- Task: (1) **Click** the **New Query** tab:  -- (2) **Copy** n **Paste** the below **TSQL** code into a new **Query Window.**  -- (3) Click on E**xecute** to run the **TSQL**: |
| SELECT \* FROM Albums. |

If your table contains a lot of data, you should probably limit the results by using Top 1000 or some other number.