SQL DOWN UNDER

# SQL SERVER™ MANAGEMENT STUDIO TIPS AND TRICKS

DR GREG LOW

Updated March 2019

# SQL Server™ Management Studio Tips and Tricks

Dr Greg Low

SQL Down Under Pty Ltd

@greglow

http://ssmsbook.sqldownunder.com

First edition April 2018
Update March 2019



Cover Image by the amazing **Glen Carrie** (c/- Unsplash https://unsplash.com/photos/JiSkHnWLo2o)



The SSMS product described in this eBook is itself subject to constant change. The tips and tricks described could change at any point in time. We've done our best to make this eBook as error free as possible at the time of publication, but we don't promise that it is error free or that anything we describe will work for you.

**Note from the author:**

I've worked with SQL Server™ for decades and have compiled this list of tips and tricks over that time. This eBook is a compilation of a series of blog posts that I have either made or are scheduled to be made as part of my Thursday "Shortcut" series, for shortcuts that apply to SSMS. (On each heading, I've added the publication date and depending upon when you read this, some might be in the future).

While I'd love to remember who first showed me each one of these, the original source of any particular tip or trick is now too difficult to determine, and often may have come from multiple sources. Though, I need to thank the Data Platform MVP community for endless inspiration, and for no doubt being the ones who showed me many of these items over the years. For that reason, I consider myself the editor more than the author.

We intend to keep enhancing and upgrading this book. If you have feedback for it, please send that to ssmsbook@sqldownunder.com


Need to learn about SQL Server ? SQL Down Under offer online on-demand courses that you can take whenever you want.

You can learn with Greg right now !

We're rapidly expanding our list of courses. And we have courses on topics that others don't even consider.

Check us out now at http://training.sqldownunder.com

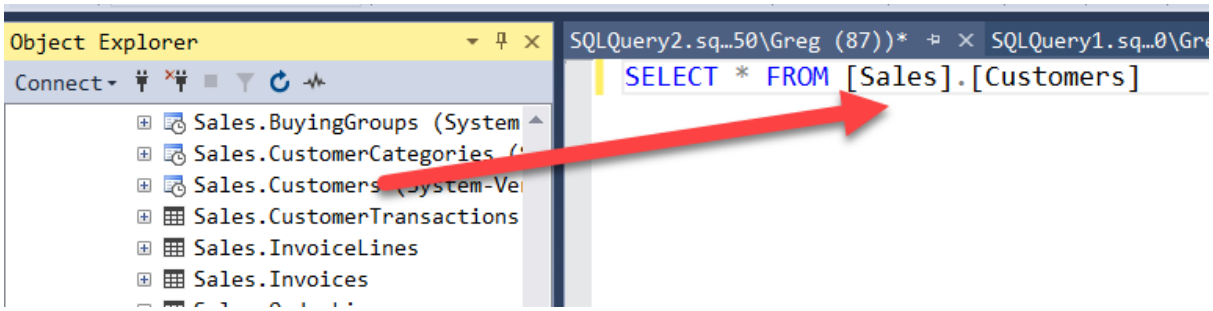
# Contents

# 1 Object Explorer

## 1.1 Dragging all column names from Object Explorer

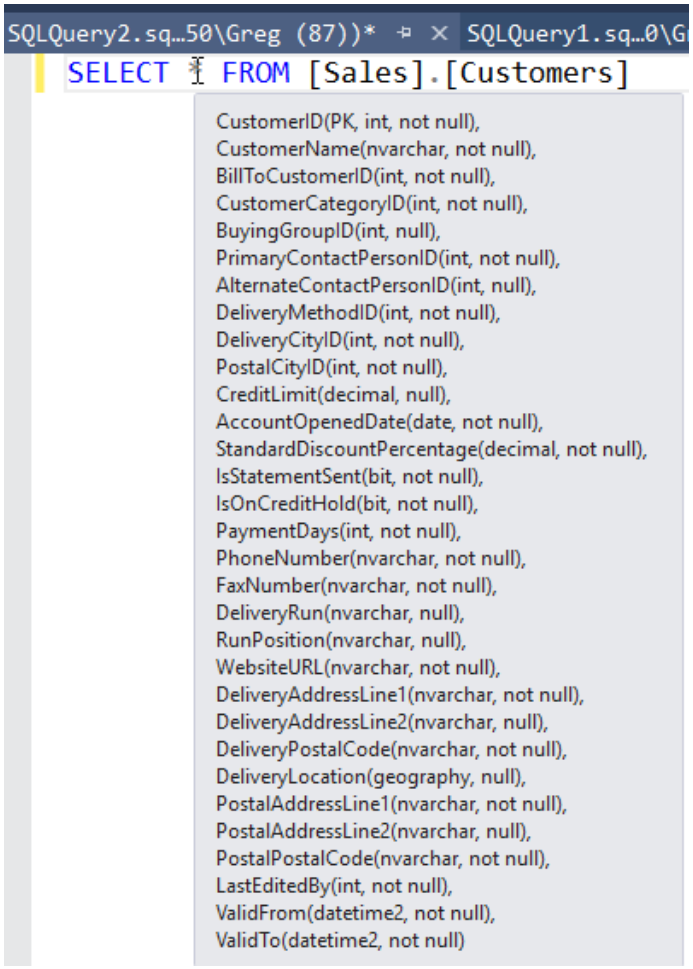
This is a really popular shortcut in SQL Server Management Studio (SSMS) but I continue to be amazed how many people aren't aware of it.

Object Explorer is a very useful part of SSMS and you can drag pretty much any name that you see in it, across to a query window.

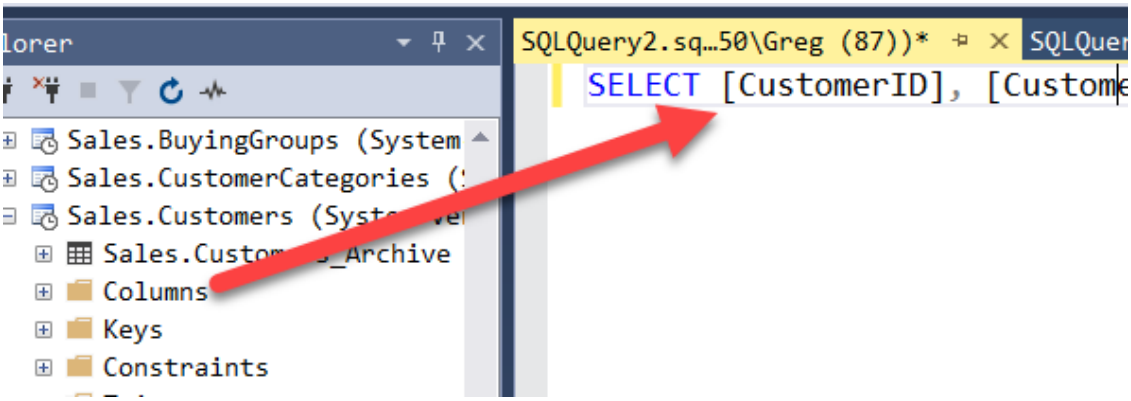



You could do the same for each column in the Columns list.

You might also realize that you can hover over the asterisk and see a list of columns:

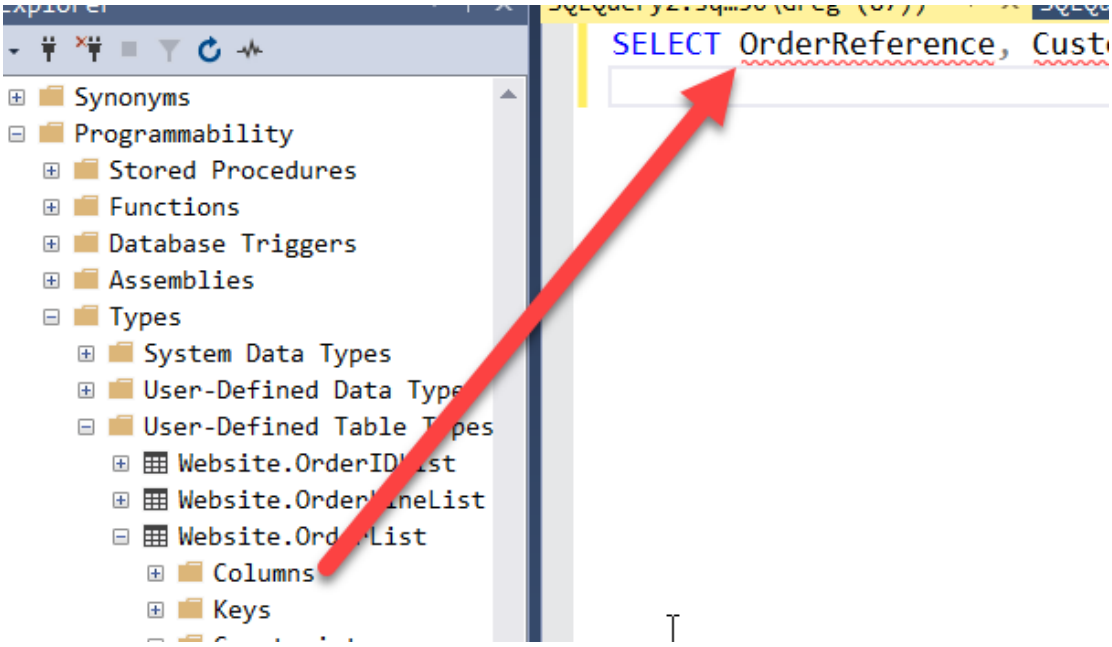


But the one that many people aren't aware of is that you can drag the word **Columns** across and get all the columns:




The final thing that seemed missing about this for me, was that I often wanted to do the same with the column list in a user defined table type.

I asked @sqltoolsguy and the SSMS team nicely, and only a few weeks later, that works too:




In earlier versions, the quoting wasn't consistent. Notice that the table type column names aren't quoted, but the table columns are. Version 17.6 and later have that fixed.

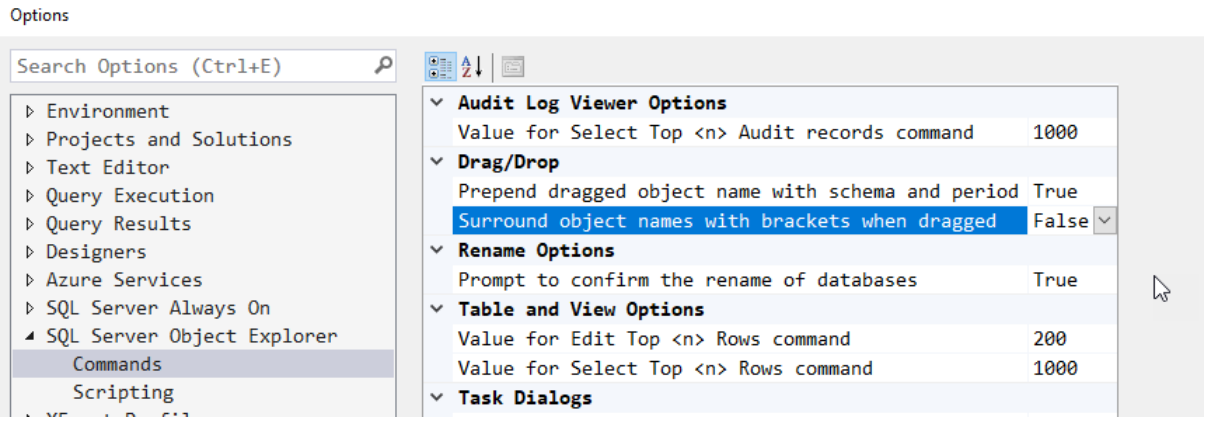
## 1.2 Control the use of square brackets (quoting) when dragging columns

Previously I showed how useful it is to be able to drag columns from either a table or from user-defined table type onto the query surface.

But one of the first comments I hear from people who did that is:

I hate all those square brackets. How do I stop that?

And prior to version 17.6, there wasn't a good answer to that question. In 17.6, the following option was added to **Tools> Options > SQL Server Object Explorer > Commands**:

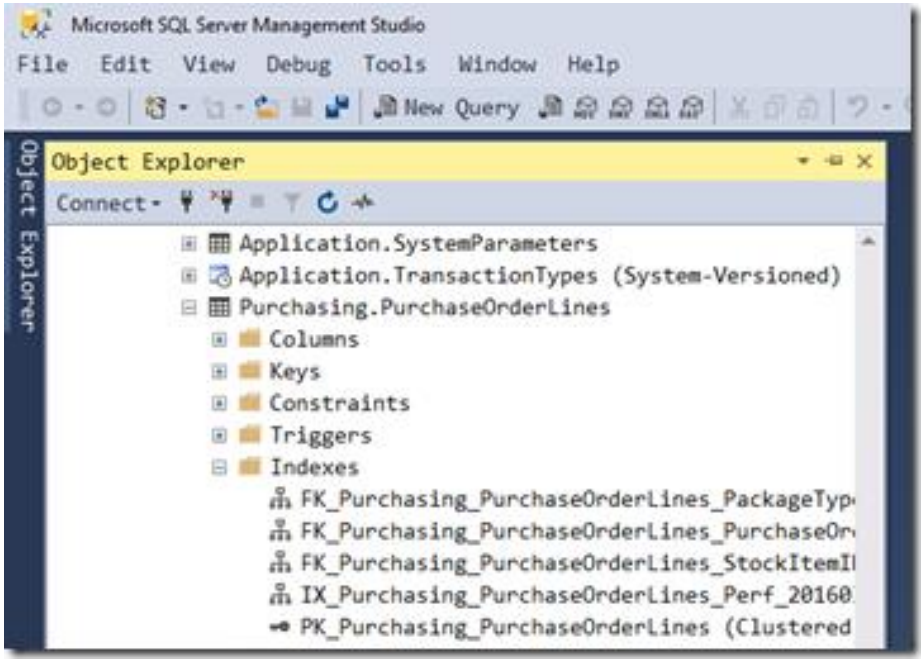



Bonus points for the team because they also provided us with an option to decide whether schema names are dragged as well. That's mostly used for tables but also applies to other schema-bound objects like stored procedures.

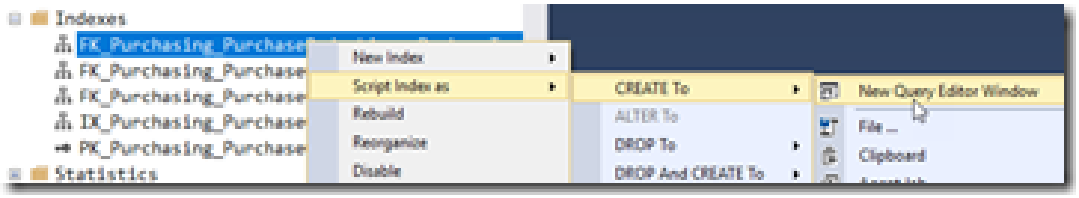
## 1.3 Script Multiple Objects at Once in SSMS

If I want to script all the indexes on a table (or all the tables, all the stored procedures, etc, etc.), yes you can do that the long way by using the Generate Scripts option but there's a better way.

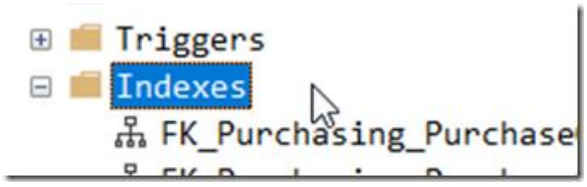
Let's use the Purchasing.PurchaseOrderLines table from WideWorldImporters as an example. Here are the indexes on it:



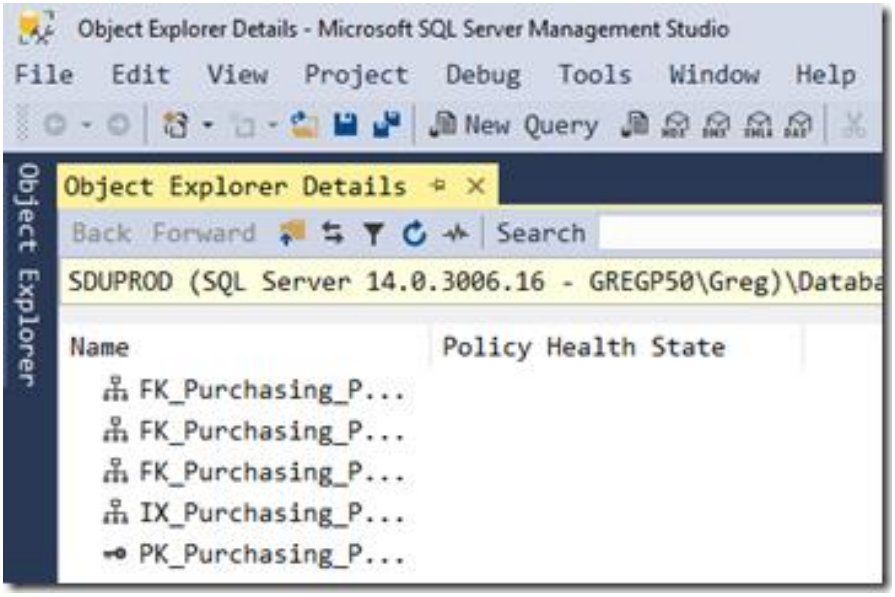

The scripting options are well-known. You right-click the object, and can navigate to the scripting option:



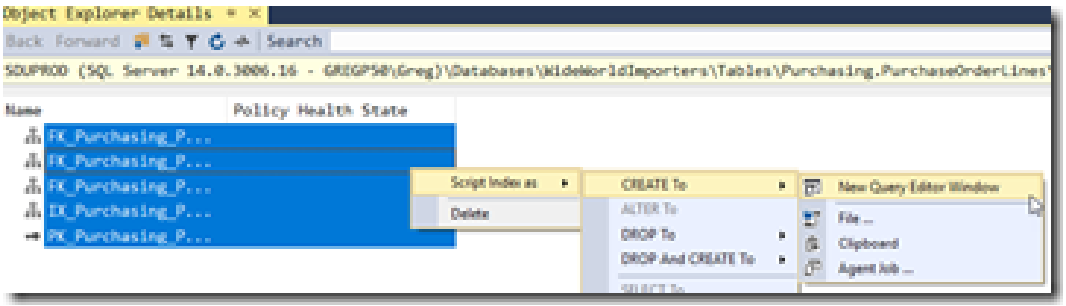
But this is tedious when you want to script a whole set of them. Instead, start by clicking on the **Indexes** node itself:




When it's selected, click **F7** to open the **Object Explorer Details** pane:




Then click the top item, shift-click the bottom item (to select them all), and finally right-click to see that you can script all of them at once:

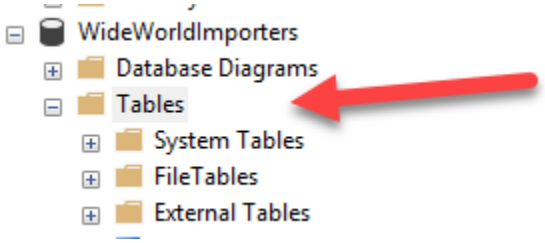


## 1.4 Add columns to Object Explorer Details window (ie: rowcounts of tables)

I've mentioned in another section about scripting multiple objects at once, how useful the Object Explorer Details window is, and how little understood it is.

Another useful option in it, is that the displayed columns can be changed. In particular, you can add columns that would be useful. Let's look at an example.

In Object Explorer, I've expanded the WideWorldImporters database and clicked on the word Tables:

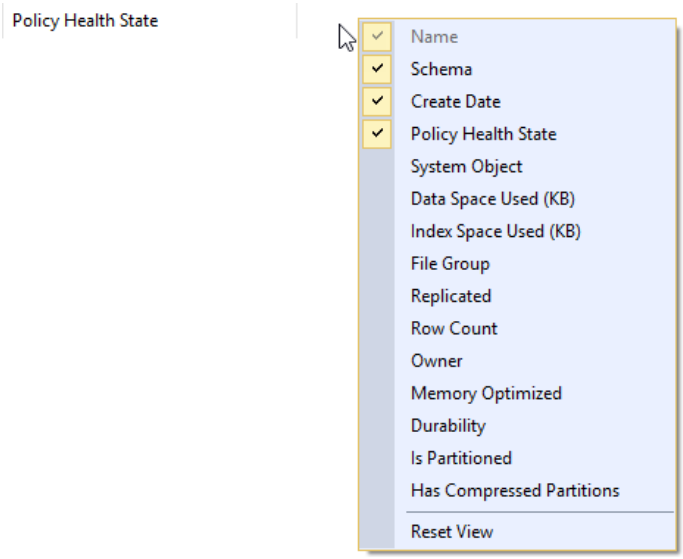



Next, I hit the **F7** key, and the **Object Explorer Details** pane opens showing this:

| Name | Schema | Create Date | Policy Health State |
|---|---|---|---|
| System Tables | | | |
| FileTables | | | |
| External Tables | | | |
| Cities (System-Versioned) | Application | 2/06/2016 10:04 AM | |
| Countries (System-Versioned) | Application | 2/06/2016 10:04 AM | |
| DeliveryMethods (System-Versioned) | Application | 2/06/2016 10:04 AM | |
| PaymentMethods (System-Versioned) | Application | 2/06/2016 10:04 AM | |
| People (System-Versioned) | Application | 2/06/2016 10:04 AM | |
| StateProvinces (System-Versioned) | Application | 2/06/2016 10:04 AM | |
| SystemParameters | Application | 2/06/2016 10:04 AM | |
| TransactionTypes (System-Versioned) | Application | 2/06/2016 10:04 AM | |
| PurchaseOrderLines | Purchasing | 2/06/2016 10:04 AM | |
| PurchaseOrders | Purchasing | 2/06/2016 10:04 AM | |
| SupplierCategories (System-Versioned) | Purchasing | 2/06/2016 10:04 AM | |
| Suppliers (System-Versioned) | Purchasing | 2/06/2016 10:04 AM | |
| SupplierTransactions | Purchasing | 2/06/2016 10:04 AM | |
| BuyingGroups (System-Versioned) | Sales | 2/06/2016 10:04 AM | |
| CustomerCategories (System-Versioned) | Sales | 2/06/2016 10:04 AM | |
| Customers (System-Versioned) | Sales | 2/06/2016 10:04 AM | |
| CustomerTransactions | Sales | 2/06/2016 10:04 AM | |
| InvoiceLines | Sales | 2/06/2016 10:04 AM | |
| Invoices | Sales | 2/06/2016 10:04 AM | |
| OrderLines | Sales | 2/06/2016 10:04 AM | |
| Orders | Sales | 2/06/2016 10:04 AM | |
| SpecialDeals | Sales | 2/06/2016 10:04 AM | |
| ColdRoomTemperatures (System-Versioned) | Warehouse | 6/06/2016 11:06 AM | |
| Colors (System-Versioned) | Warehouse | 2/06/2016 10:04 AM | |
| PackageTypes (System-Versioned) | Warehouse | 2/06/2016 10:04 AM | |
| StockGroups (System-Versioned) | Warehouse | 2/06/2016 10:04 AM | |
| StockItemHoldings | Warehouse | 2/06/2016 10:04 AM | |
| StockItems (System-Versioned) | Warehouse | 2/06/2016 10:04 AM | |
| StockItemStockGroups | Warehouse | 2/06/2016 10:04 AM | |
| StockItemTransactions | Warehouse | 2/06/2016 10:04 AM | |
| VehicleTemperatures | Warehouse | 6/06/2016 11:06 AM | |

I get a list of tables showing Name, Schema, Create Date, and Policy Health State. Policy was an interesting concept that I thought never got fully baked into the product, so it's not of great interest to me, and really is just clutter. However, the number of rows in the table would be much more interesting.

If I right-click on the heading row, I get these options:




Now we're talking. I'll remove the Policy Health State, and Create Date (I don't value it too much most of the time either), and add Data and Index space used, and Row Count. And we're left with a much more useful result:

| Name | Schema | Data Space Used (KB) | Index Space Used (KB) | Row Count |
|---|---|---|---|---|
| System Tables | | | | |
| FileTables | | | | |
| External Tables | | | | |
| Cities (System-Versioned) | Application | 3960 | 896 | 37940 |
| Countries (System-Versioned) | Application | 1744 | 72 | 190 |
| DeliveryMethods (System-Versioned) | Application | 8 | 24 | 10 |
| PaymentMethods (System-Versioned) | Application | 8 | 24 | 4 |
| People (System-Versioned) | Application | 624 | 312 | 1111 |
| StateProvinces (System-Versioned) | Application | 384 | 64 | 53 |
| SystemParameters | Application | 8 | 40 | 1 |
| TransactionTypes (System-Versioned) | Application | 8 | 24 | 13 |
| PurchaseOrderLines | Purchasing | 1248 | 832 | 8367 |
| PurchaseOrders | Purchasing | 136 | 192 | 2074 |
| SupplierCategories (System-Versioned) | Purchasing | 8 | 24 | 9 |
| Suppliers (System-Versioned) | Purchasing | 8 | 120 | 13 |
| SupplierTransactions | Purchasing | 248 | 688 | 2438 |
| BuyingGroups (System-Versioned) | Sales | 8 | 24 | 2 |
| CustomerCategories (System-Versioned) | Sales | 8 | 24 | 8 |
| Customers (System-Versioned) | Sales | 304 | 344 | 663 |
| CustomerTransactions | Sales | 8944 | 12536 | 97147 |
| InvoiceLines | Sales | 41552 | 13192 | 228265 |
| Invoices | Sales | 90856 | 14128 | 70510 |
| OrderLines | Sales | 38240 | 34536 | 231412 |
| Orders | Sales | 5504 | 6104 | 73595 |
| SpecialDeals | Sales | 8 | 88 | 2 |
| ColdRoomTemperatures (System-Versioned) | Warehouse | 0 | 1 | 4 |
| Colors (System-Versioned) | Warehouse | 8 | 24 | 36 |
| PackageTypes (System-Versioned) | Warehouse | 8 | 24 | 14 |
| StockGroups (System-Versioned) | Warehouse | 8 | 24 | 10 |
| StockItemHoldings | Warehouse | 16 | 16 | 227 |
| StockItems (System-Versioned) | Warehouse | 112 | 128 | 227 |
| StockItemStockGroups | Warehouse | 16 | 48 | 442 |
| StockItemTransactions | Warehouse | 2720 | 37648 | 236667 |
| VehicleTemperatures | Warehouse | 32033 | 6980 | 65998 |

And, I hear you ask: "do I have to do this every time?" The answer is no. If you close the Object Explorer Details Window and hit F7 on the Tables node again, the same output appears.

## 1.5 Extended Properties for objects

I started working with SQL Server in 1992, but all through the 1980's and 1990's, I was also working with Progress 4GL. I thought it was the best of the character-based 4GLs but unfortunately, they did a poor job of migrating to Windows and we decided to stop using the product.

One thing that I used to love with Progress though is that the metadata for each column in the database was much richer than what is present in SQL Server. In fact, Microsoft Access was probably closer to it in that regard. It's something I really missed when moving to SQL Server. In Progress, when I defined a column, I could also define things like:

- The name that would be displayed as a prompt on an input screen for this column
- The format that data in this column would be displayed in
- Any non-default input format or masking
- And so on

Having this type of information in the database and stored in a consistent form can greatly reduce the amount of boilerplate code that needs to be written in applications.

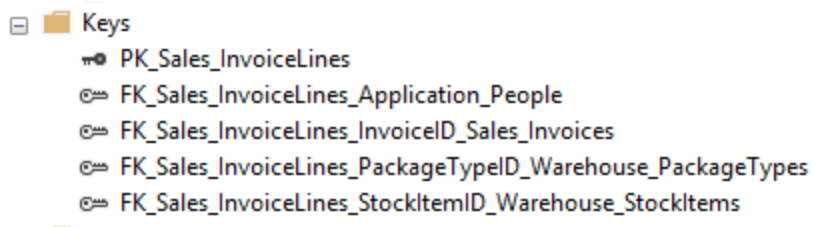
SQL Server does have a concept of extended properties but what I think is missing is a set of "well-known" properties such as "Description". The SQL Server team is starting to use these properties now for things like classifying (or labelling) data for security purposes.

I'd like to see them used more often. Here's an example:

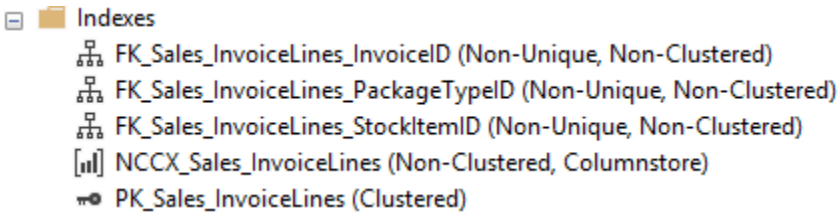
I often come across indexes in databases and when I ask why the index was created, no-one knows why. That also means that they are scared to remove the index. How much easier would this type of thing be if we had standard properties for each index that described why it was added?

We can already do this, but I just wish there were agreed standards for this.

As an example though, I tend to name indexes that are just present to support foreign keys, with the same name as the foreign key. That then matches how primary key indexes are named. In the WideWorldImporters database, you can see the naming convention. For the Sales.InvoiceLines table, here are the keys:

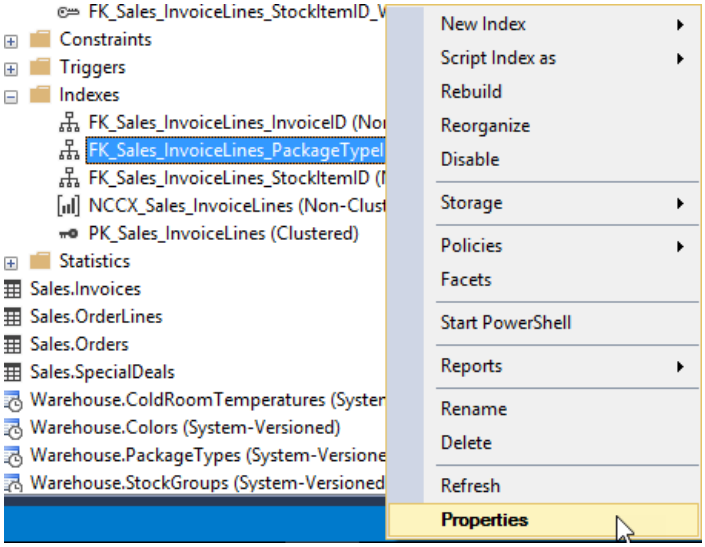



And here are the indexes:

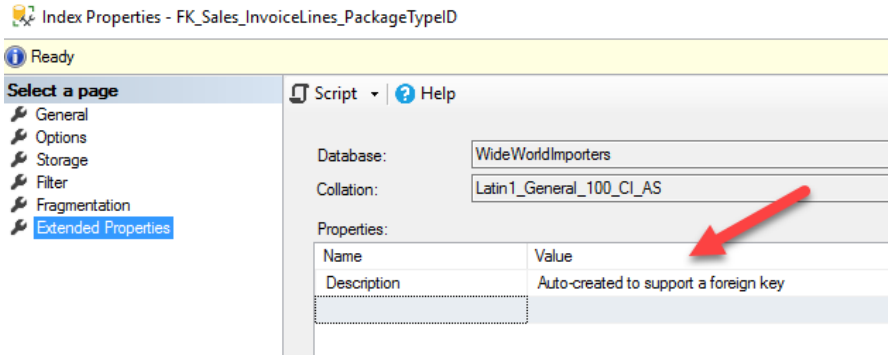



It's pretty obvious which indexes are there to support foreign keys. (Note we made a conscious decision not to index the Application.People related foreign key).

But for other indexes, how would you know why they are there? We included that info in our code generation, by using extended properties. To see this, right-click one of these indexes, and choose Properties:




On the Extended Properties page, you can see a description of why this index was created:

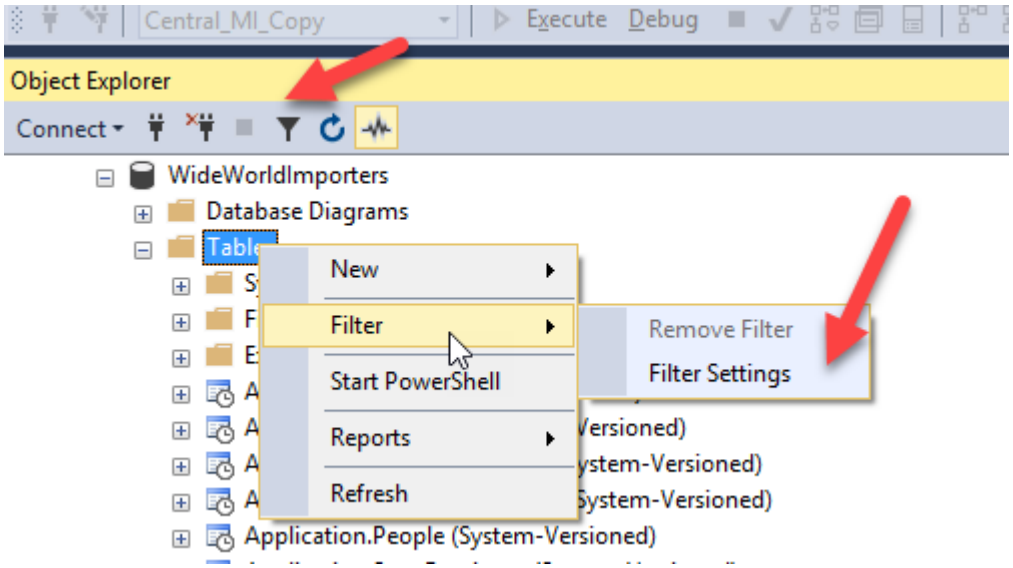



I'd like to see much richer metadata for each object in SQL Server and I suspect we might have to do that with a set of extended properties. I'd like to see a standard set of these defined.

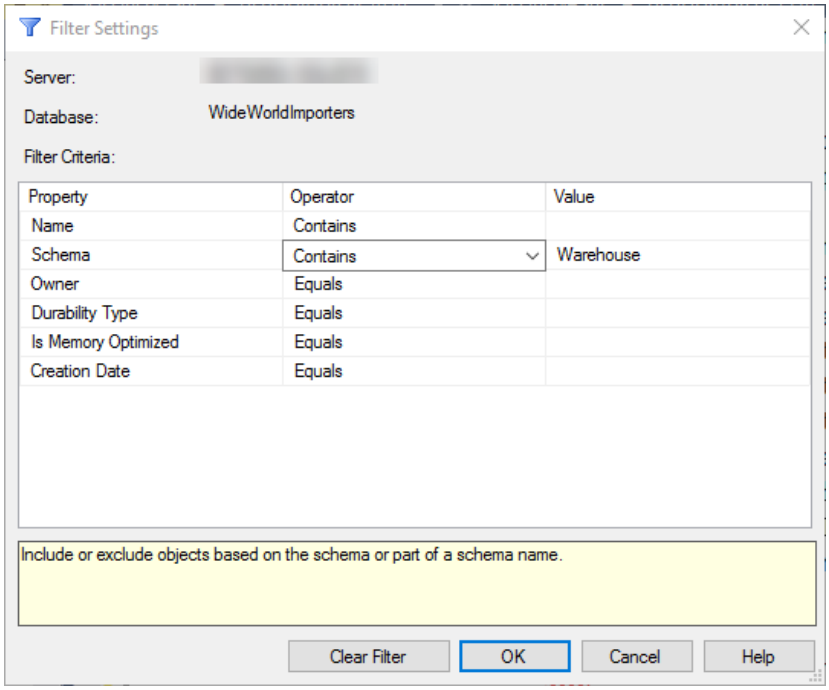
## 1.6 Filters in Object Explorer

If you are working with databases with large numbers of objects, the contents of Object Explorer can start to become a bit overwhelming. I have to admit that I don't understand why it doesn't offer an option to group by schema. That would be helpful.

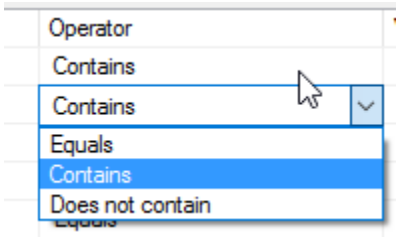
But you can at least filter by things like schema, when you need to work with a specific set of objects. You'll notice that if you click on the database name, that the filter in the toolbar is grayed out, but if you click on a node below that like Tables, you can click on the toolbar filter icon. You can also right-click the node and choose to filter:



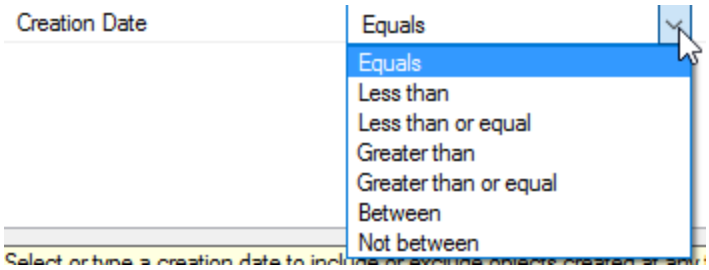

You then are presented with these options:



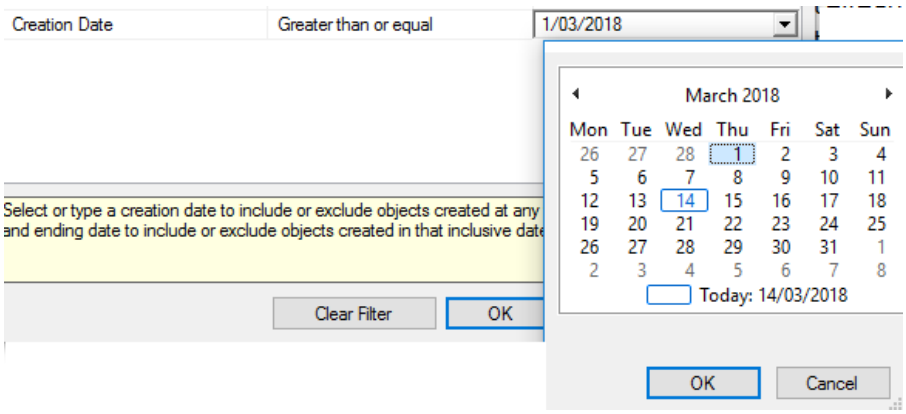

That will work to give me just the tables in the Warehouse schema but notice that the operator is Contains. There are other options:




So I could see all the tables except those that are in this schema. Note that Creation Date has even more options:




I can then also use this for other interesting queries such as “Just show me tables that have been created since 1st March:

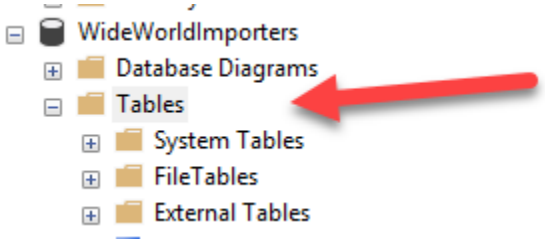


### 1.7 Navigate as you type in sorted OED pane

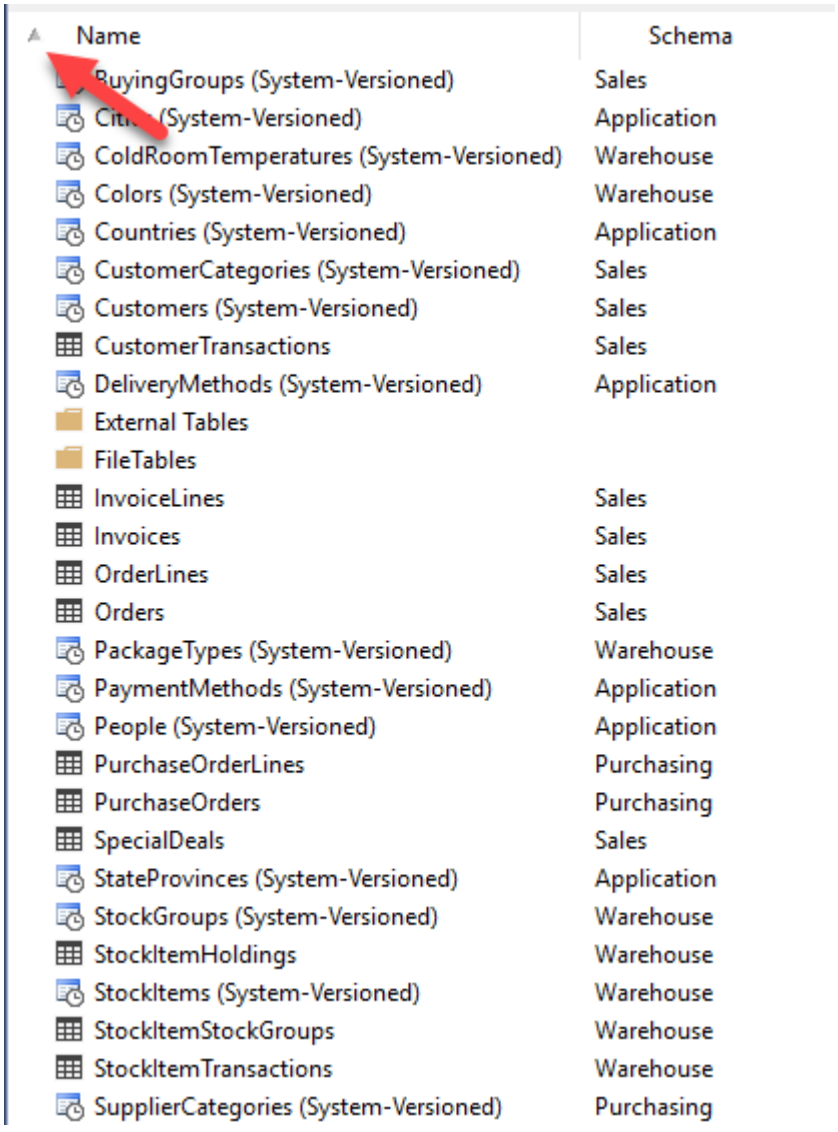
I've mentioned a number of times how useful I think the Object Explorer Details panel is in SQL Server Management Studio.

Another option in that panel that might not be so obvious is the sorted navigation. Here's an example.

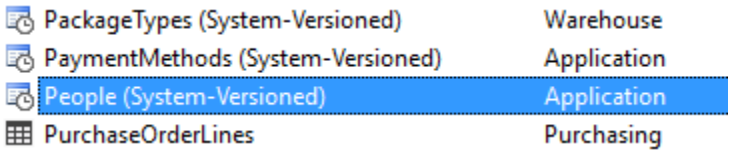
I've opened WideWorldImporters in Object Explorer, and clicked on the Tables node:




I then hit F7 to open the Object Explorer Details pane, and click the Name heading to sort the table list:




Then if I type (say the letters Pe), you'll notice that I'm positioned immediately to the first table starting with Pe.



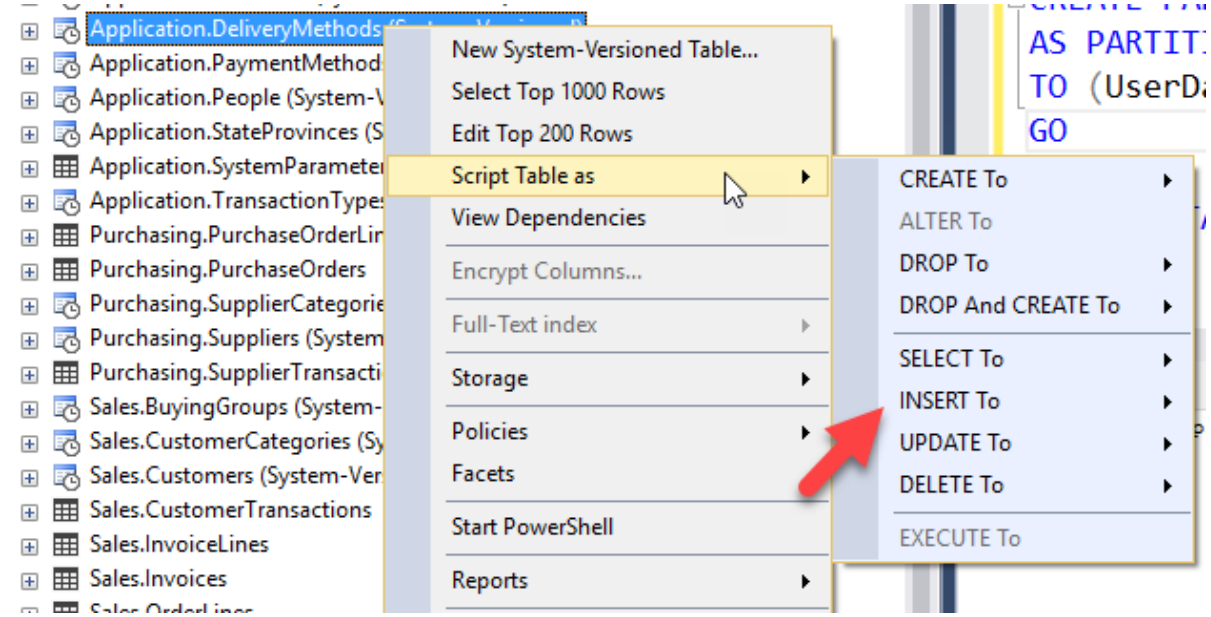

When you have a small number of tables, this is no big deal, but when you have a lot of tables, this is very useful.

## 1.8 Generate insert scripts (script data)

Over the years, I've had a surprising number of questions on how to output all the data in a table as a series of INSERT statements.

SQL Server Management Studio has had the ability to do this for a long time. Here's an example.

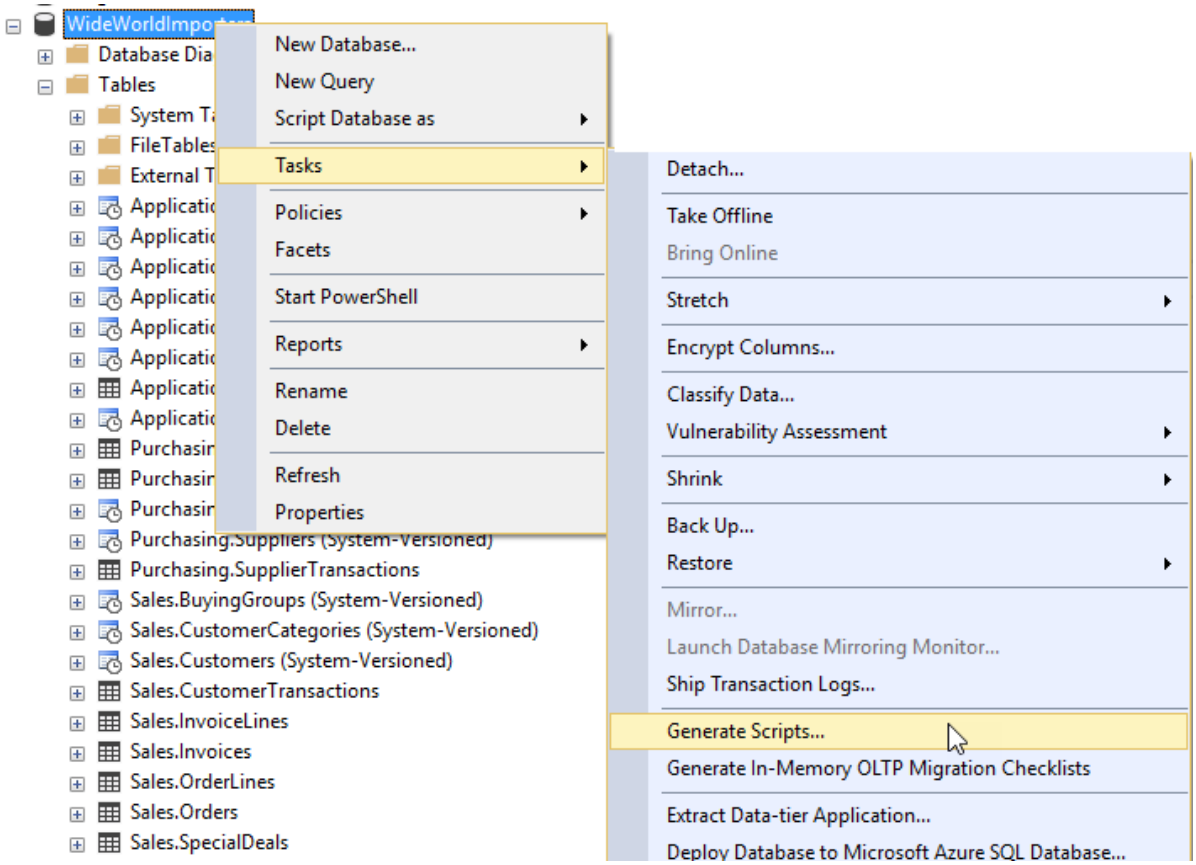
In Object Explorer, I've expanded the WideWorldImporters database, then expanded Tables. Where people come unstuck is they right-click the table, and look at the scripting options:



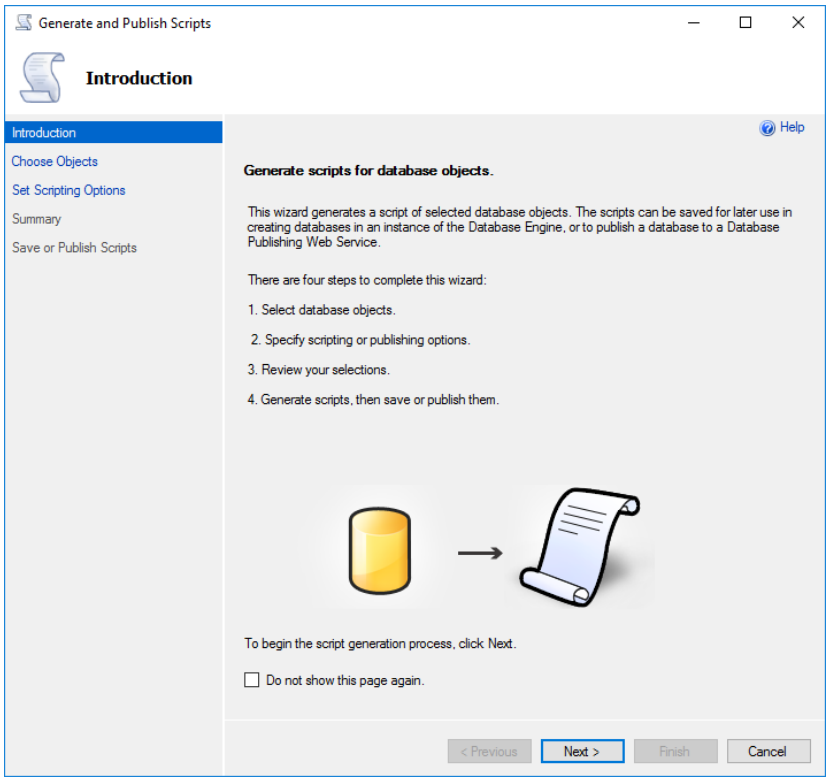

But if you use the option to script INSERT to a new query window, you get this:

```
USE [WideWorldImporters]
GO

INSERT INTO [Application].[DeliveryMethods]
           ([DeliveryMethodID]
           ,[DeliveryMethodName]
           ,[LastEditedBy])
     VALUES
           (<DeliveryMethodID, int,>
           ,<DeliveryMethodName, nvarchar(50),>
           ,<LastEditedBy, int,>)
GO
```

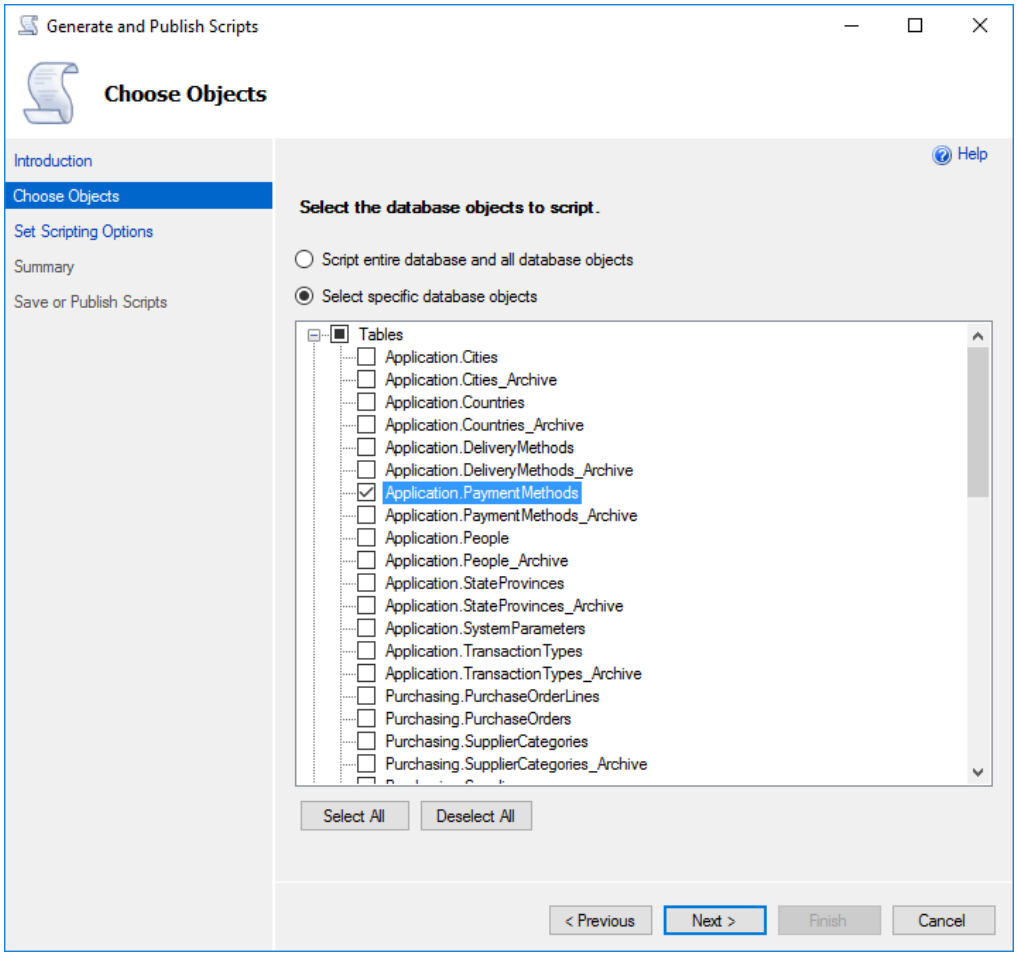
It's scripting an INSERT statement, not the data as a set of INSERT statements. Now, I think that option should actually be present here, but the way to get to it, is a bit more roundabout. You need to right-click the database, then choose **Tasks**, then **Generate Scripts**.



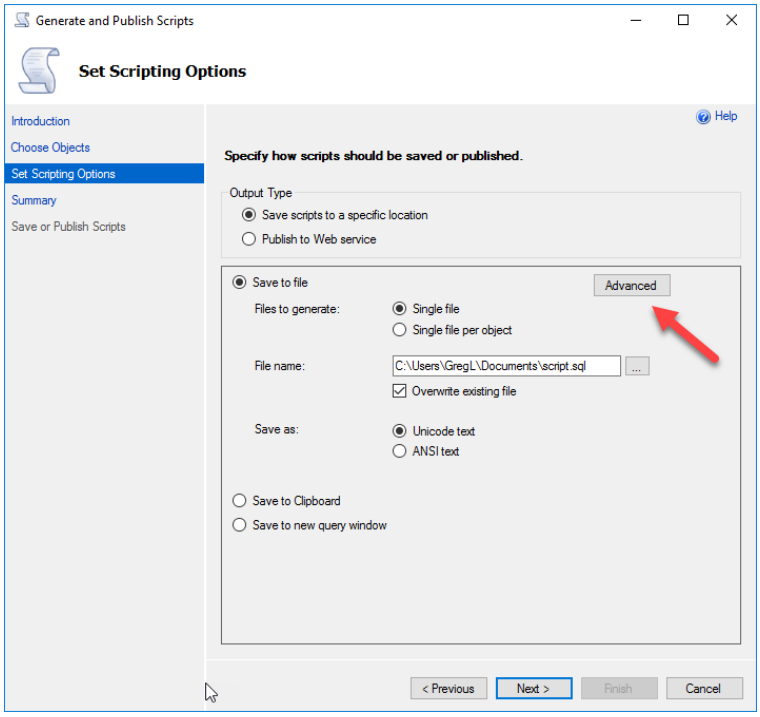

On the first window, click **Next**.



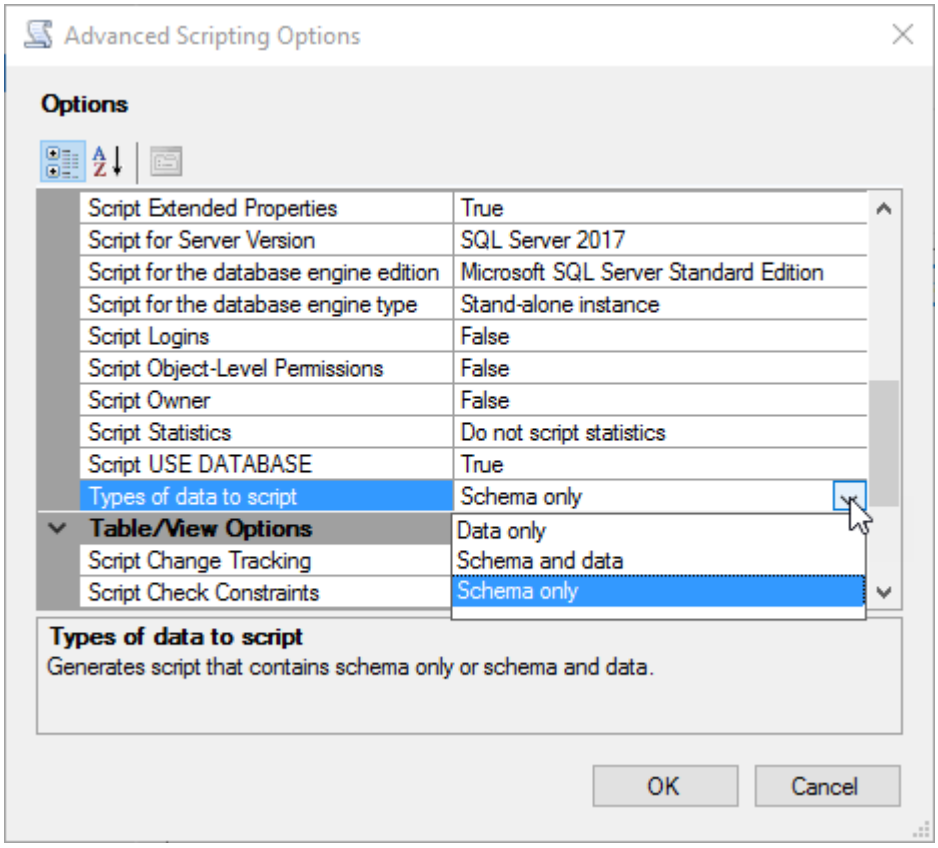
On the second screen, choose **Select specific database objects**, then expand **Tables**, and pick the table you're after, then click **Next**:



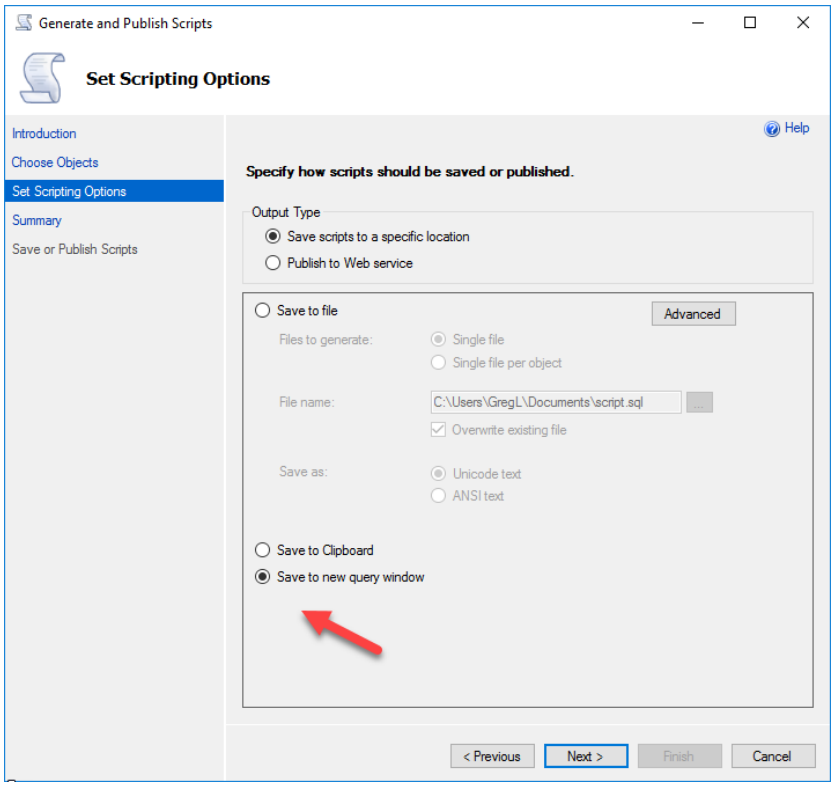

On the third screen, click the **Advanced** button.



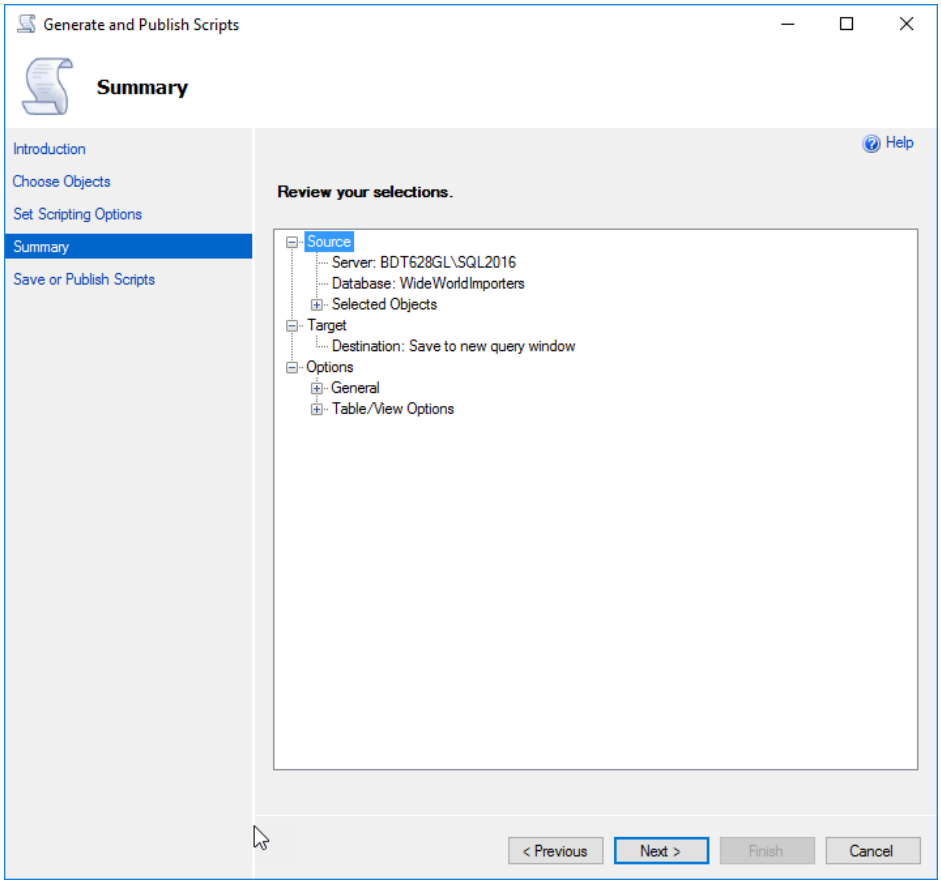
In the **Advanced Scripting Options** window, scroll down to find **Types of data to script** and note the options:



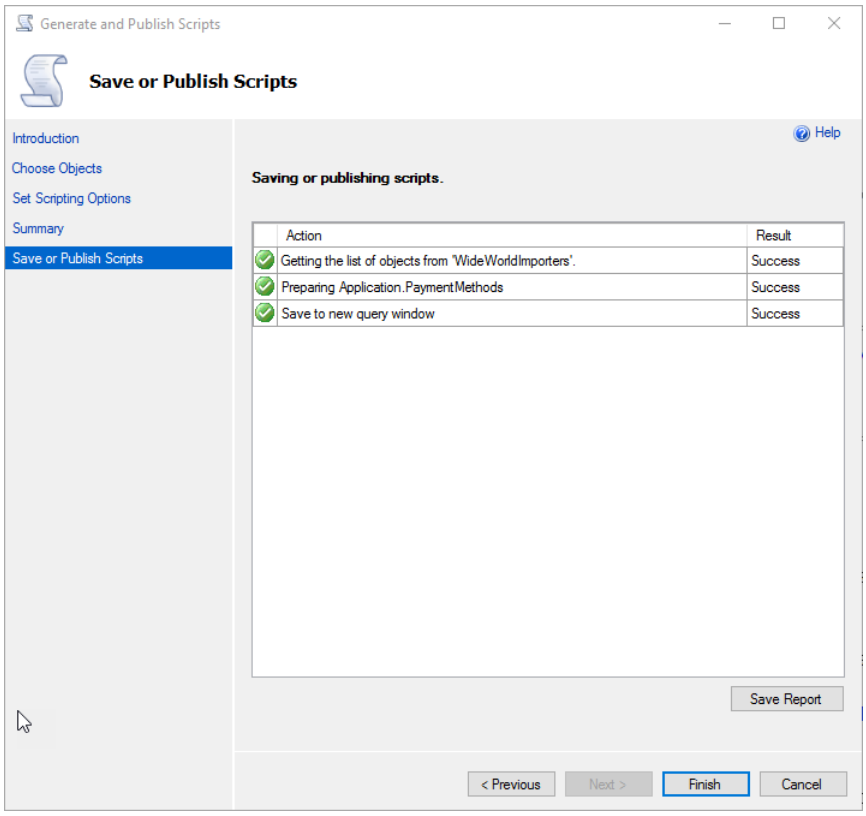

Choose **Data only** if that's all you want, then **OK**. I've then chosen **Save to new query window**, and **Next**.



I've then reviewed what's going to happen and clicked **Next** yet again.




On the final screen, I've noted the generation happening, then clicked **Finish**.



Then the window I was after has appeared:

```
USE [WideWorldImporters]
GO
INSERT [Application].[PaymentMethods] ([PaymentMethodID], [PaymentMethodName], [LastEditedBy], [V
GO
INSERT [Application].[PaymentMethods] ([PaymentMethodID], [PaymentMethodName], [LastEditedBy], [V
GO
INSERT [Application].[PaymentMethods] ([PaymentMethodID], [PaymentMethodName], [LastEditedBy], [V
GO
INSERT [Application].[PaymentMethods] ([PaymentMethodID], [PaymentMethodName], [LastEditedBy], [V
GO
```

Clearly I wish this was much simpler to do as it's a fairly common operation, certainly more common than many of the operations that are on the right-click context menu for a table. It's also not going to be suitable for large amounts of data but often that's not what you need to script.

And you might then want to use a SQL formatting tool to clean up the output window.

## 1.9 Turn off option to prevent saving changes that require table re-creation

I don't use the table designer in SQL Server Management Studio. Sorry, but I just don't like it, or the options that it chooses for me. I'd rather use T-SQL every time, but I'm also the first to admit that there are plenty of people who would use that designer. And when they do, many run into an error that says:

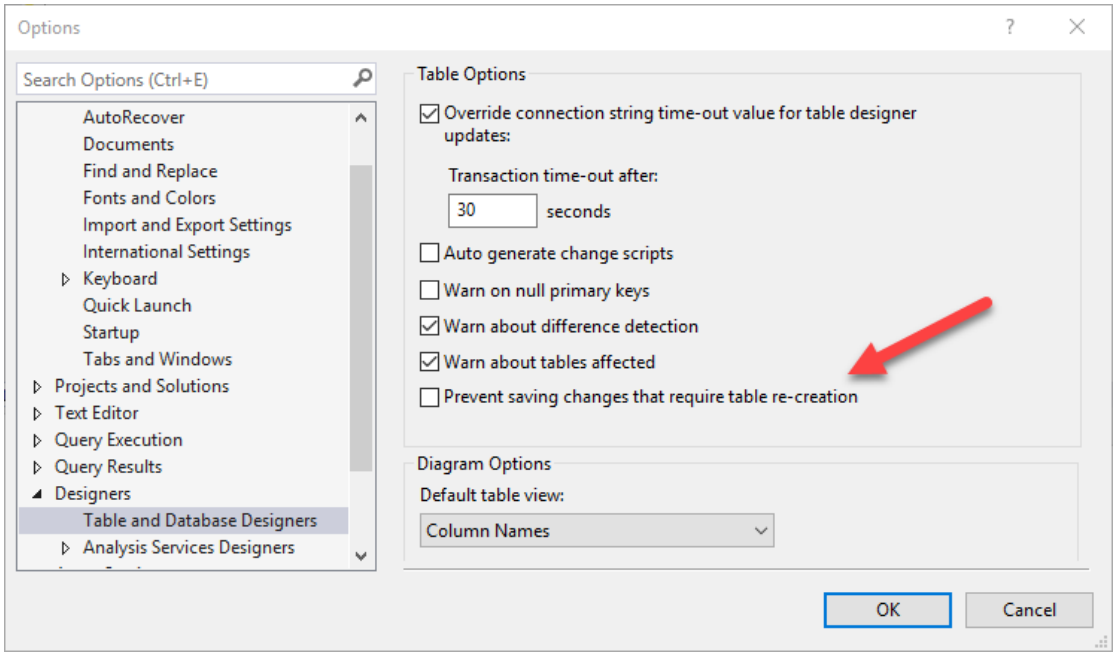
**Saving changes is not permitted. The changes that you have made require the following tables to be dropped and re-created. You have either made changes to a table that can't be re-created or enabled the option Prevent saving changes that require the table to be re-created.**

And they throw up their hands in horror. According to Books Online, this is typically because they've done one of these types of things:

Change the Allow Nulls setting for a column
Reorder columns in the table
Change the column data type
Add a new column

Having to recreate a table for things like changing a column's data type seems way too heavy-handed. SQL Server Management Studio takes a safe option here. You can override it by doing this:

In Tools > Options > Designers > Table and Database Designers




While you can turn that off, be very careful if you decide to do so. You can cause issues, particularly if you are also using Change Tracking. (Existing tracking information is deleted when the table is recreated). You could also affect DDL triggers and many other things. For simple situations, this might help.

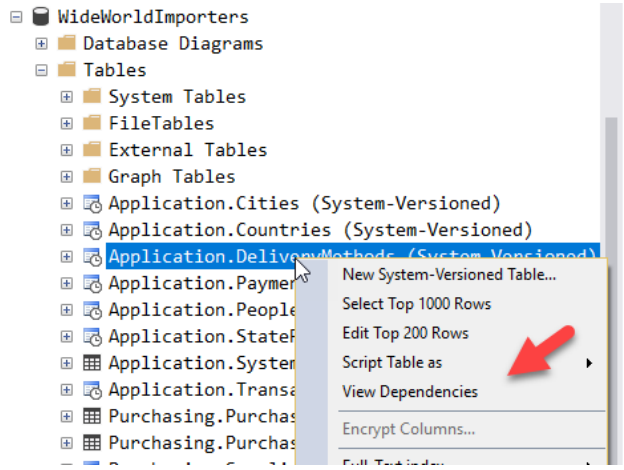
**As I mentioned, if it was me, I'd use T-SQL statements that do just the actions that I need.**

## 1.10 Dependency tracking

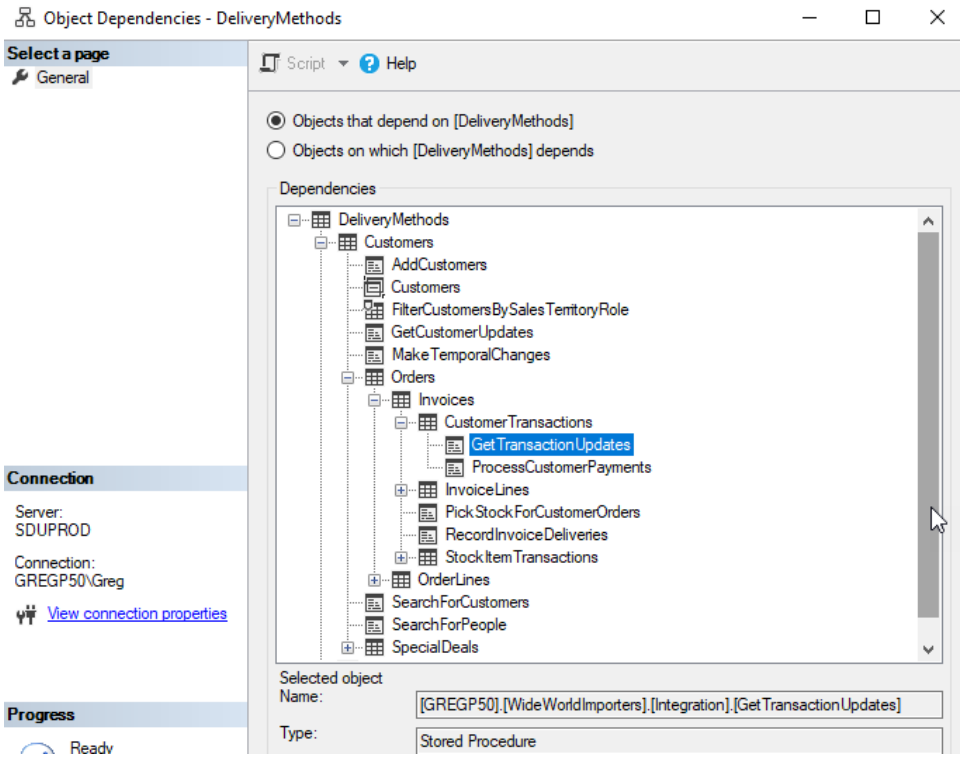
In early versions of SQL Server, the only way to try to track dependencies between tables, procedures, functions, etc. was to use the sp_depends stored procedure. And everyone thought it lied. The real problem was that it didn't understand partial dependencies and deferred resolution of objects. For example, it got confused if you created a procedure that mentioned a table, then later created the table.

SQL Server 2012 introduced far superior dependency views, and SQL Server Management Studio (SSMS) now shows dependencies using those views under the covers.

Here's an example. If I right-click the Application.DeliveryMethods table in the WideWorldImporters database, I can choose to View Dependencies:

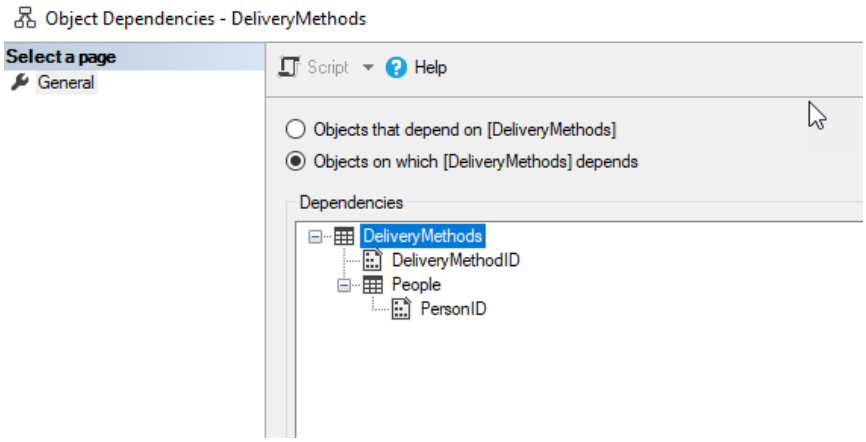



By default, you are shown the objects that depend upon the selected object (in this case the table that we right-clicked):



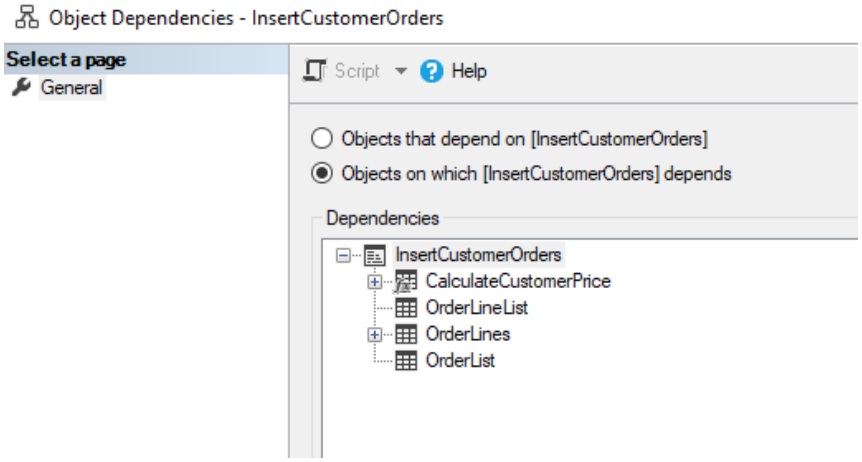
Note that this is a multi-layer dependency tree. We can see that the Customers table depends upon this table, as does the AddCustomers stored procedure. The Orders table also depends upon the DeliveryMethods table, and through that, the CustomerTransactions table, and from there, onto the GetTransactionUpdates procedure.

We can also see objects that the DeliveryMethods table depends upon:




In this case, we can see that the table depends upon its own primary key (ie: DeliveryMethodID), and on the People table because there is a foreign key to the PersonID column in that table, for the last person who modified the rows in the table.

We can also see dependencies for other types of objects. Here is the tree for the InsertCustomerOrders stored procedure:



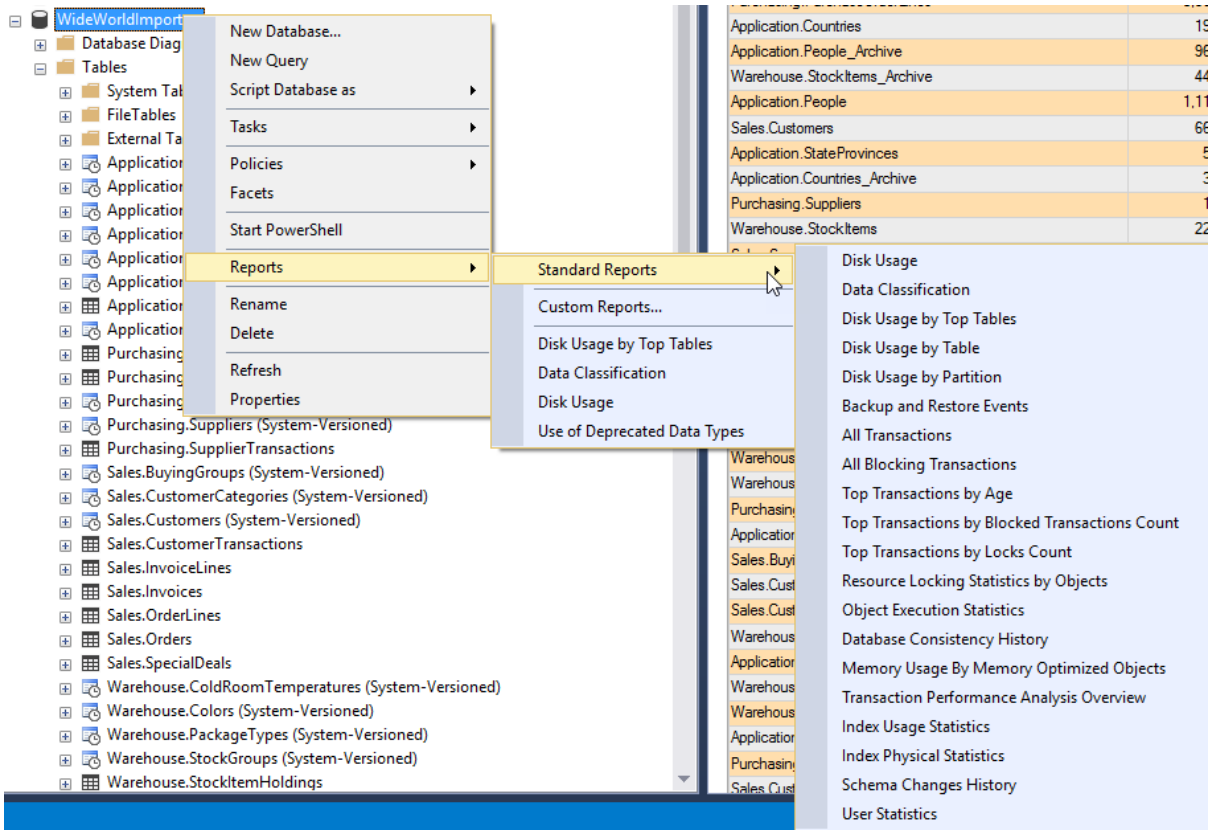

It depends upon the OrderLines table, the OrderList and OrderLineList table types, and the CalculateCustomerPrice function.

I'm really pleased that such a good dependency system is available within SSMS.

## 1.11 Built-in standard reports

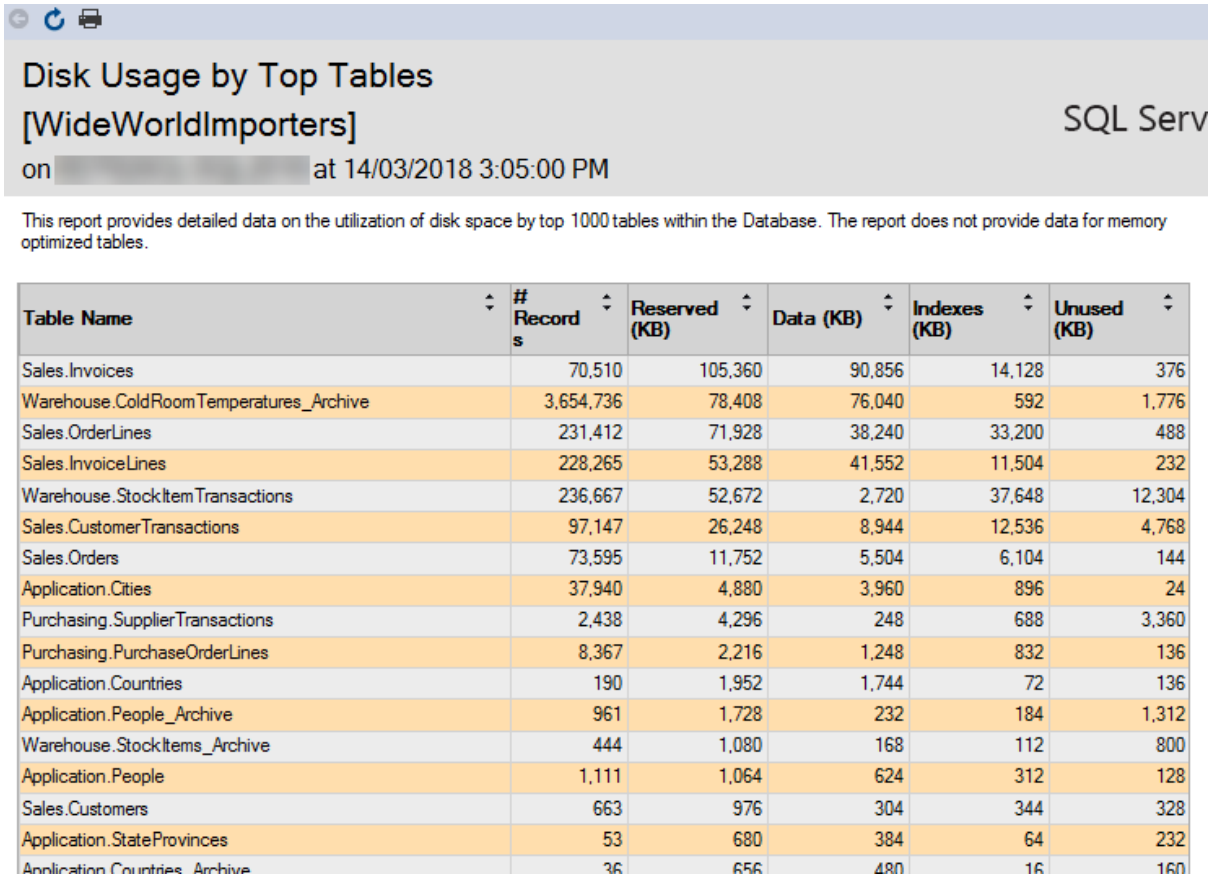
SQL Server Management Studio provides a large amount of information about databases by letting the user navigate around Object Explorer and view the properties of objects.

What I'm often surprized by though, is the number of users who haven't ever explored the reports that are available. In another section, we'll look at creating custom reports, but there is a wonderful set of built-in standard reports that you should explore. For example, if I right-click the WideWorldImporters database, I can see these reports:




In this section, I just want to highlight a few of the most useful reports.

One of the common queries when databases get larger is about what's taking up all the space. That one is easy to answer by using Disk Usage by Top Tables. (Note: this report shows the top 1000 by usage, whereas Disk Usage by Table shows all tables).



Disk Usage by Top Tables
[WideWorldImporters]
on [redacted] at 14/03/2018 3:05:00 PM

SQL Serv

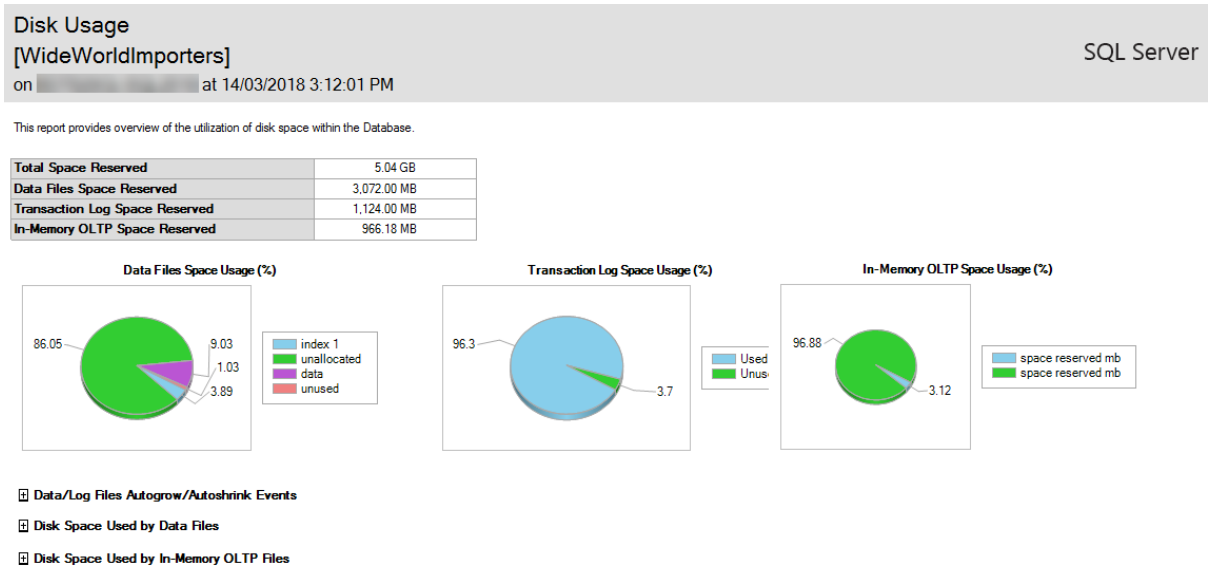
This report provides detailed data on the utilization of disk space by top 1000 tables within the Database. The report does not provide data for memory optimized tables.

| Table Name | # Records | Reserved (KB) | Data (KB) | Indexes (KB) | Unused (KB) |
|---|---|---|---|---|---|
| Sales.Invoices | 70,510 | 105,360 | 90,856 | 14,128 | 376 |
| Warehouse.ColdRoomTemperatures_Archive | 3,654,736 | 78,408 | 76,040 | 592 | 1,776 |
| Sales.OrderLines | 231,412 | 71,928 | 38,240 | 33,200 | 488 |
| Sales.InvoiceLines | 228,265 | 53,288 | 41,552 | 11,504 | 232 |
| Warehouse.StockItemTransactions | 236,667 | 52,672 | 2,720 | 37,648 | 12,304 |
| Sales.CustomerTransactions | 97,147 | 26,248 | 8,944 | 12,536 | 4,768 |
| Sales.Orders | 73,595 | 11,752 | 5,504 | 6,104 | 144 |
| Application.Cities | 37,940 | 4,880 | 3,960 | 896 | 24 |
| Purchasing.SupplierTransactions | 2,438 | 4,296 | 248 | 688 | 3,360 |
| Purchasing.PurchaseOrderLines | 8,367 | 2,216 | 1,248 | 832 | 136 |
| Application.Countries | 190 | 1,952 | 1,744 | 72 | 136 |
| Application.People_Archive | 961 | 1,728 | 232 | 184 | 1,312 |
| Warehouse.StockItems_Archive | 444 | 1,080 | 168 | 112 | 800 |
| Application.People | 1,111 | 1,064 | 624 | 312 | 128 |
| Sales.Customers | 663 | 976 | 304 | 344 | 328 |
| Application.StateProvinces | 53 | 680 | 384 | 64 | 232 |
| Application.Countries_Archive | 36 | 656 | 480 | 16 | 160 |

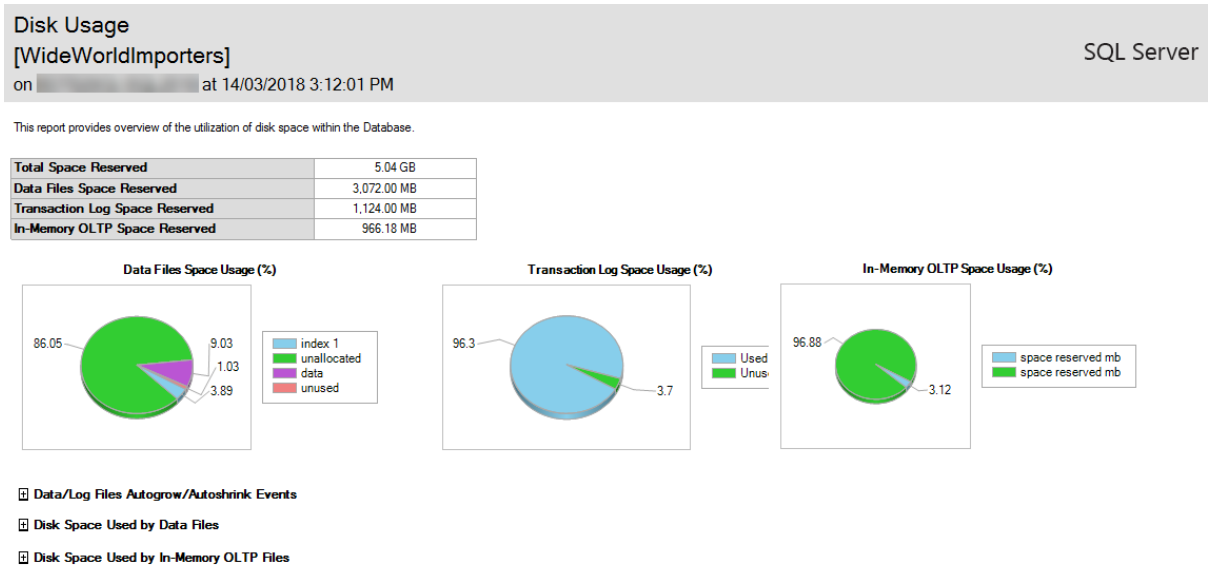
These reports are all sortable by clicking the column headings. From this, we can see that Sales.Invoices contains the most data, and by clicking the # Records heading, we can see that Warehouse.ColdRoomTemperatures_Archive is holding the most rows:

| Table Name | # Records | Reserved (KB) | Data (KB) | Indexes (KB) | Unused (KB) |
|---|---|---|---|---|---|
| Warehouse.ColdRoomTemperatures_Archive | 3,654,736 | 78,408 | 76,040 | 592 | 1,776 |
| Warehouse.StockItemTransactions | 236,667 | 52,672 | 2,720 | 37,648 | 12,304 |
| Sales.OrderLines | 231,412 | 71,928 | 38,240 | 33,200 | 488 |
| Sales.InvoiceLines | 228,265 | 53,288 | 41,552 | 11,504 | 232 |
| Sales.CustomerTransactions | 97,147 | 26,248 | 8,944 | 12,536 | 4,76 |
| Sales.Orders | 73,595 | 11,752 | 5,504 | 6,104 | 144 |
| Sales.Invoices | 70,510 | 105,360 | 90,856 | 14,128 | 376 |
| Application.Cities | 37,940 | 4,880 | 3,960 | 896 | 24 |

Another useful query is about how much free space there is, and if filegrowth events have been occurring. The standard Disk Usage report shows this:

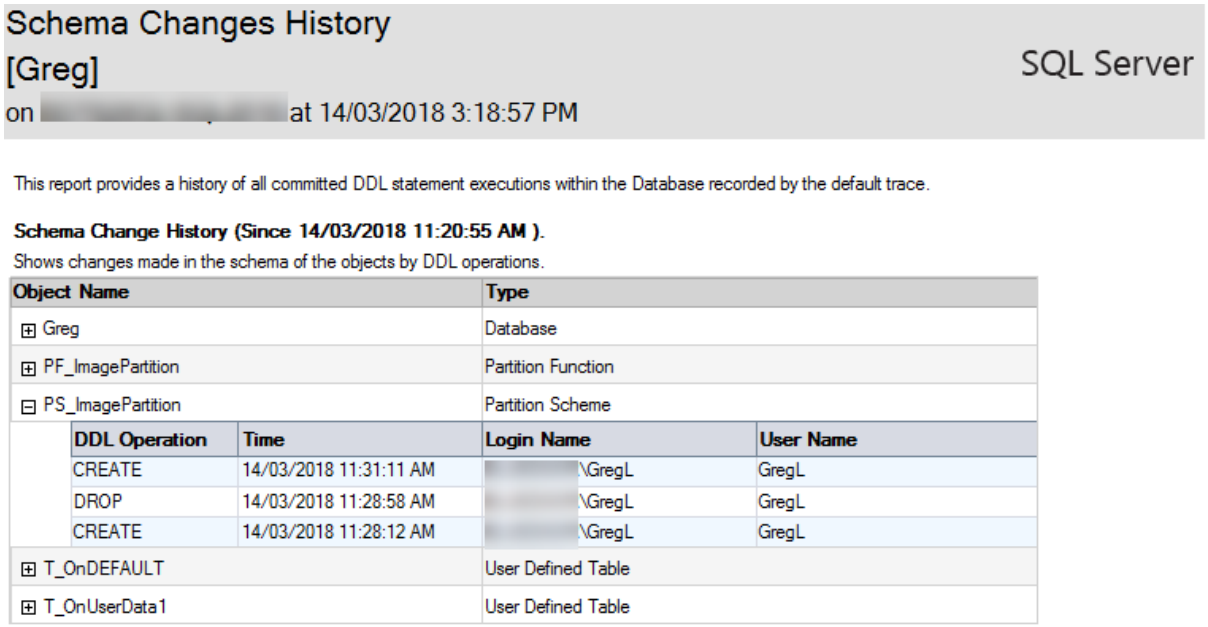



And you can expand the first line under the graphs to see autogrow or autoshrink events:




The Index Usage Statistics report puts the output of the sys.dm_db_index_usage_stats DMV into an easier-to-consume form.

Wondering who just made that recent schema change? The Schema Changes History report might help.



Schema Changes History

[Greg]

on [redacted] at 14/03/2018 3:18:57 PM

SQL Server

This report provides a history of all committed DDL statement executions within the Database recorded by the default trace.

**Schema Change History (Since 14/03/2018 11:20:55 AM ).**

Shows changes made in the schema of the objects by DDL operations.

| Object Name | Type | | |
|---|---|---|---|
| ⊞ Greg | Database | | |
| ⊞ PF_ImagePartition | Partition Function | | |
| ⊟ PS_ImagePartition | Partition Scheme | | |
| **DDL Operation** | **Time** | **Login Name** | **User Name** |
| CREATE | 14/03/2018 11:31:11 AM | [redacted]\GregL | GregL |
| DROP | 14/03/2018 11:28:58 AM | [redacted]\GregL | GregL |
| CREATE | 14/03/2018 11:28:12 AM | [redacted]\GregL | GregL |
| ⊞ T_OnDEFAULT | User Defined Table | | |
| ⊞ T_OnUserData1 | User Defined Table | | |

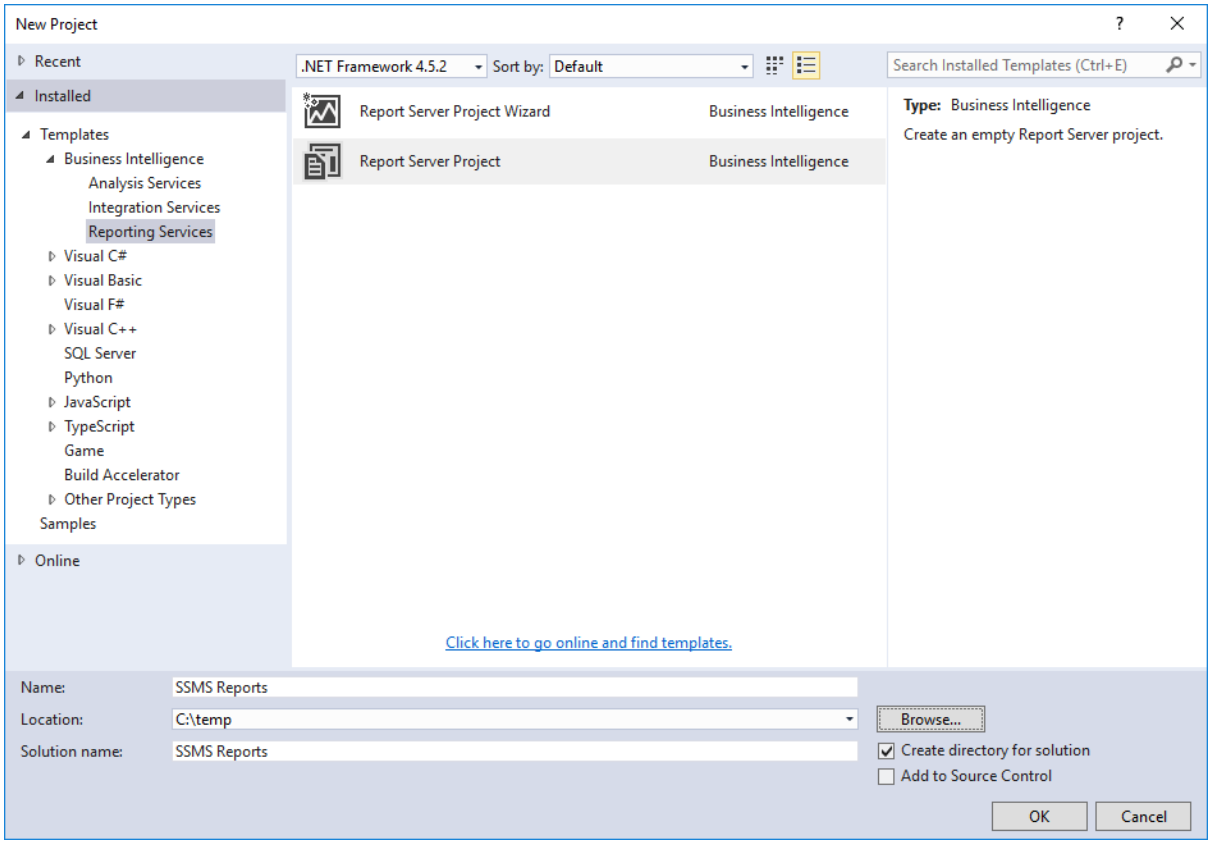
I'd encourage you to spend some time investigating what these reports offer and in the next shortcut section, we'll look at how to create custom reports.

## 1.12 Custom report creation

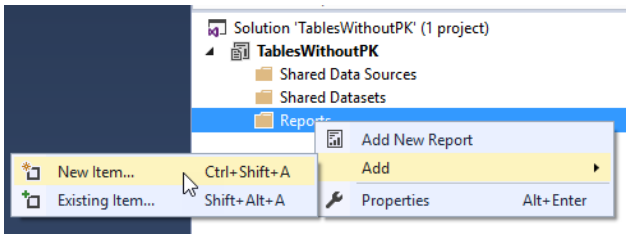
The built-in reports in SQL Server Management Studio are great but you can add your own as well. SQL Server 2005 and later have an option for Custom Reports.

Let's create a report that shows the use of deprecated data types. We'll use a stored procedure from our free SDU Tools to do that.

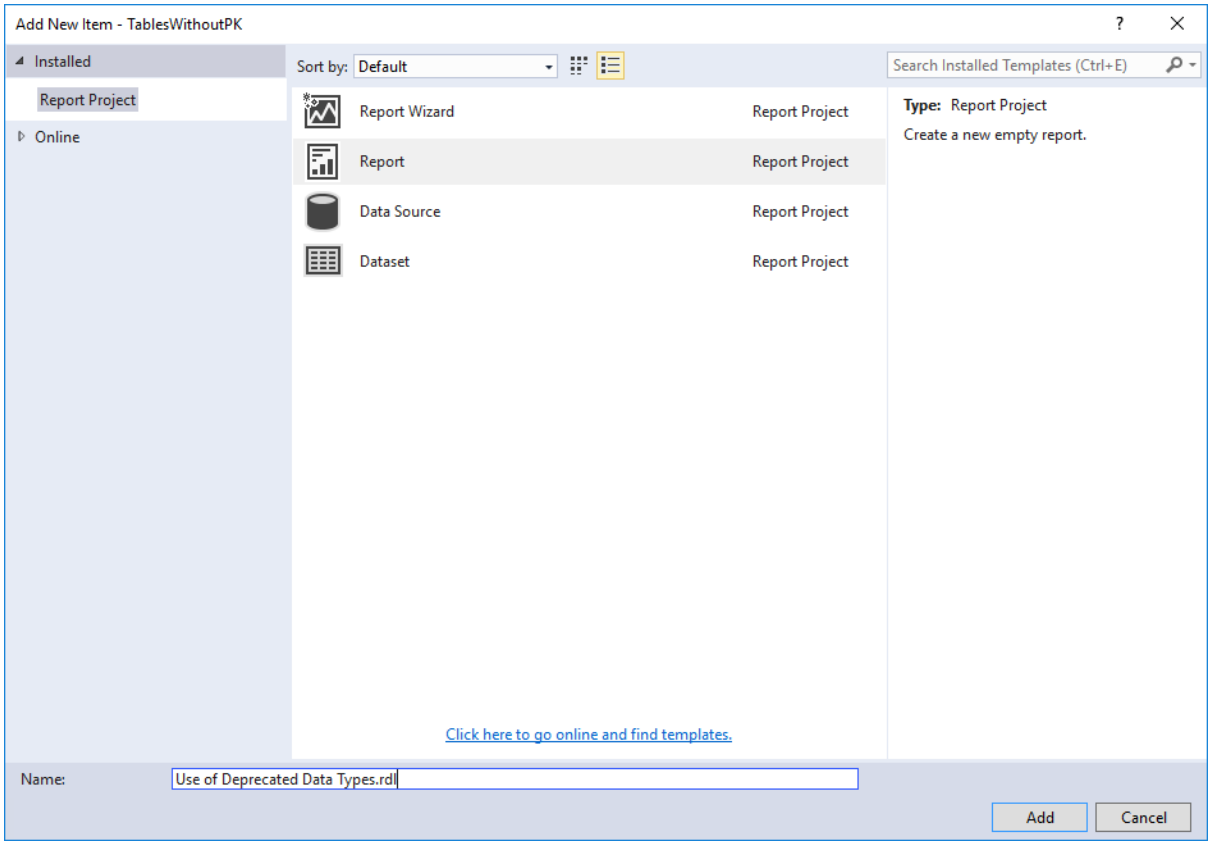
In SQL Server Data Tools, create a new Report Server project.



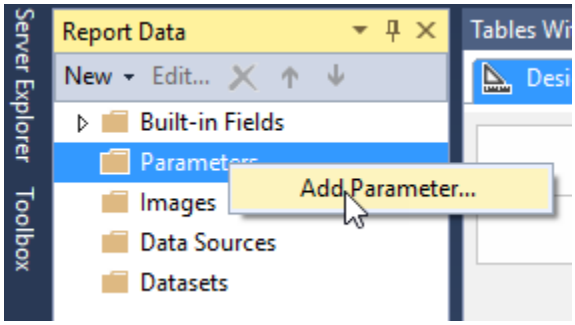

In Solution Explorer, right-click Reports, click Add, then click New Item.



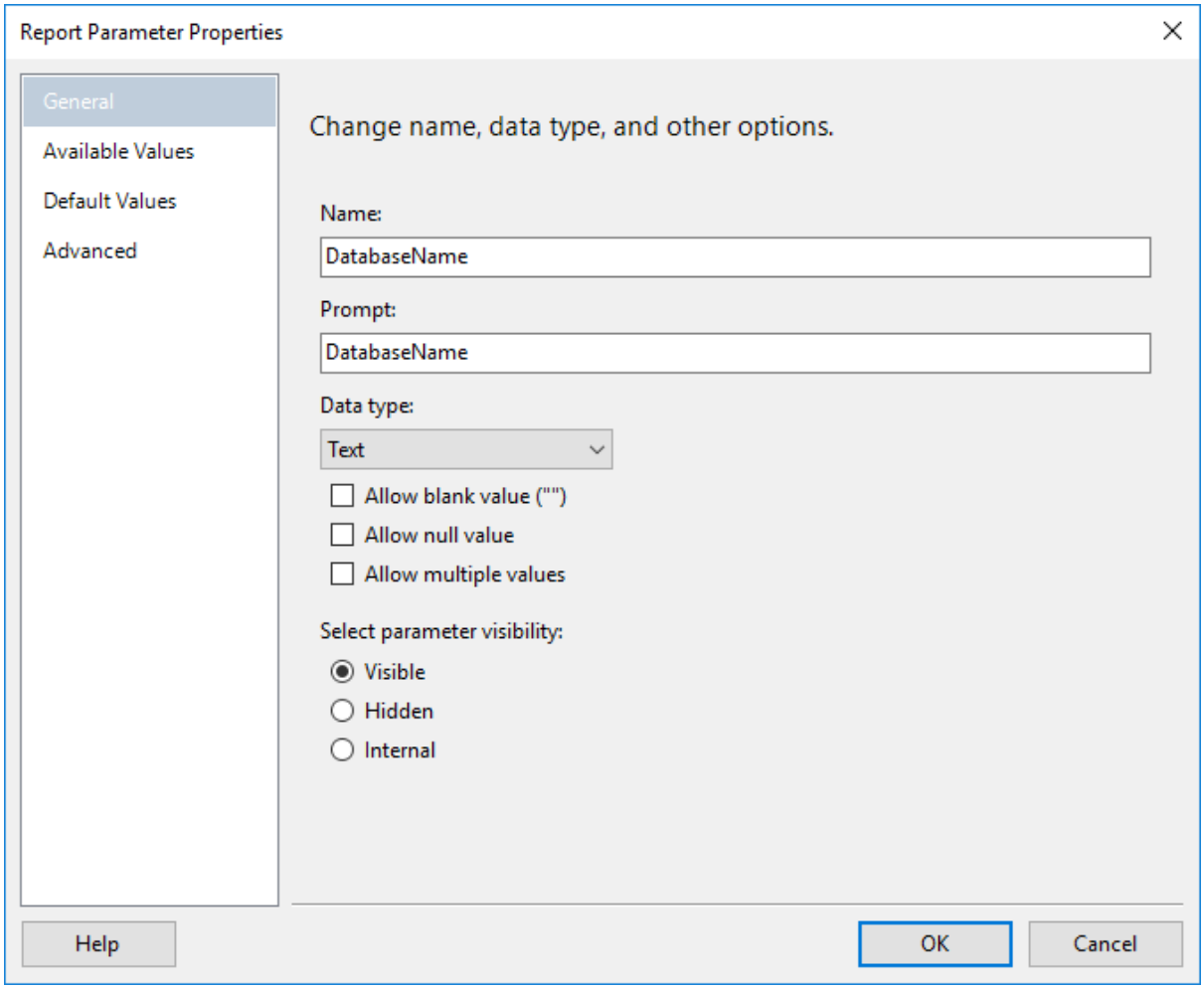

From the list of templates, choose to create a Report and give it a name.




There are a few standard parameters that are passed to your report by Object Explorer. In this case, we're only interested in the DatabaseName but we'll create them all anyway. In the Report Datra pane, right-click Parameters and click Add Parameter.



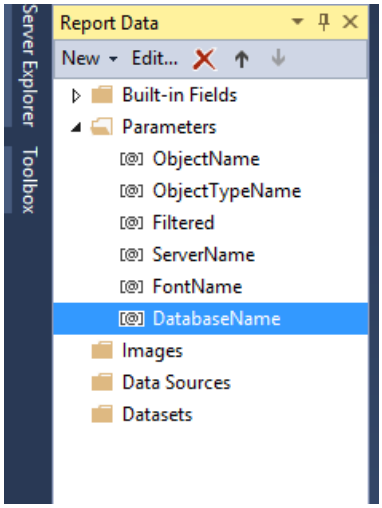

Complete the parameter details as follows. Don't choose Allow null value. We're going to require a database name.



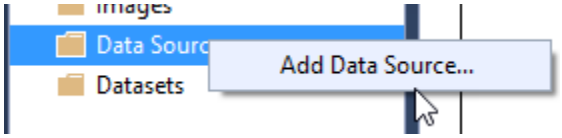

Create parameters for each of the following, ensuring that you do select Allow null value:

ObjectName (Text)
ObjectTypeName (Text)
ServerName (Text)
FontName (Text)
Filtered (Boolean)

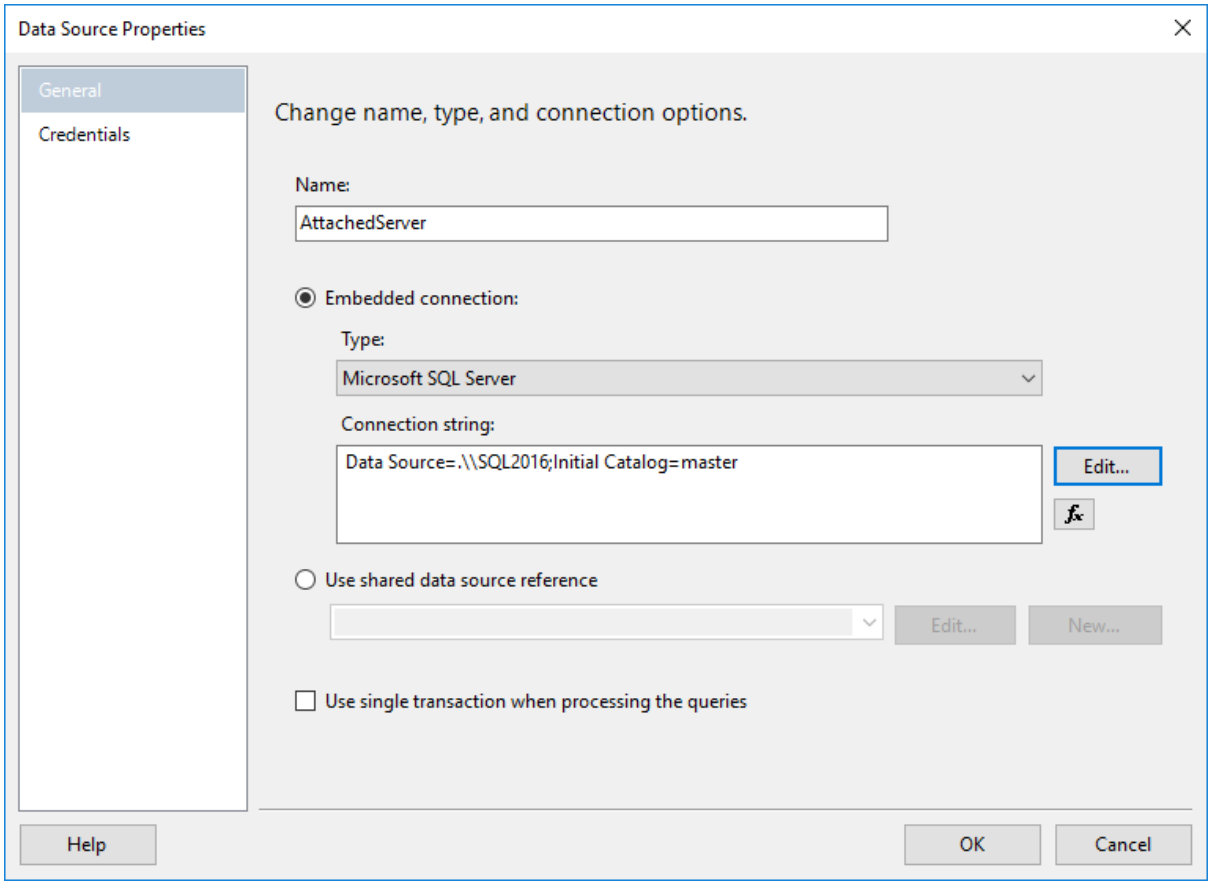
Your list should look something like this:



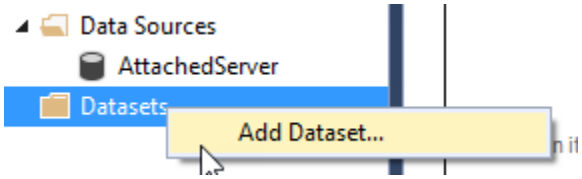

In the Report Data pane, right-click Data Sources, and click Add Data Source.



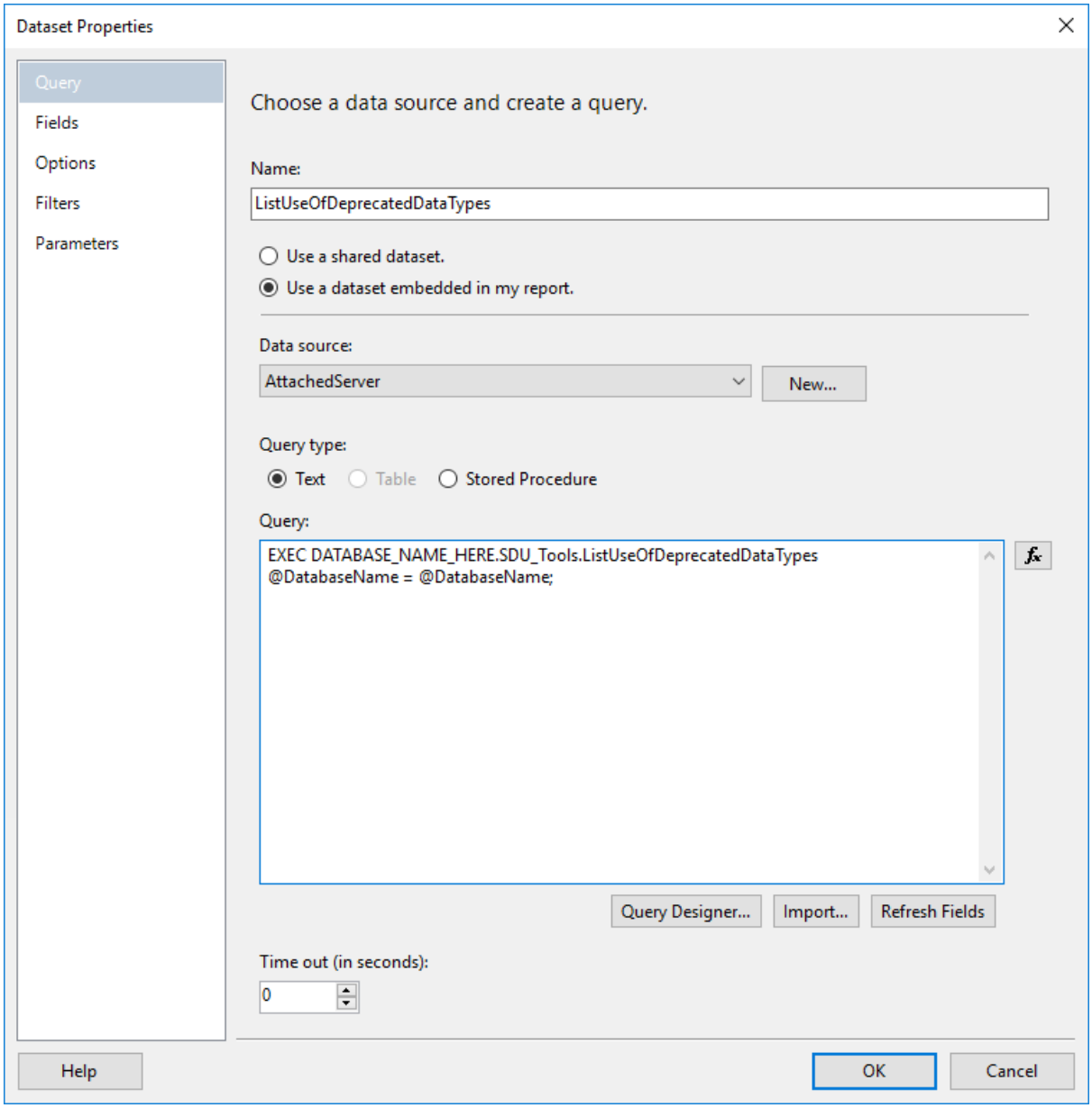

Configure the data source as follows (but note that your server connection might need to be different – mine was .\SQL2016 and is shown with a doubled backslash):



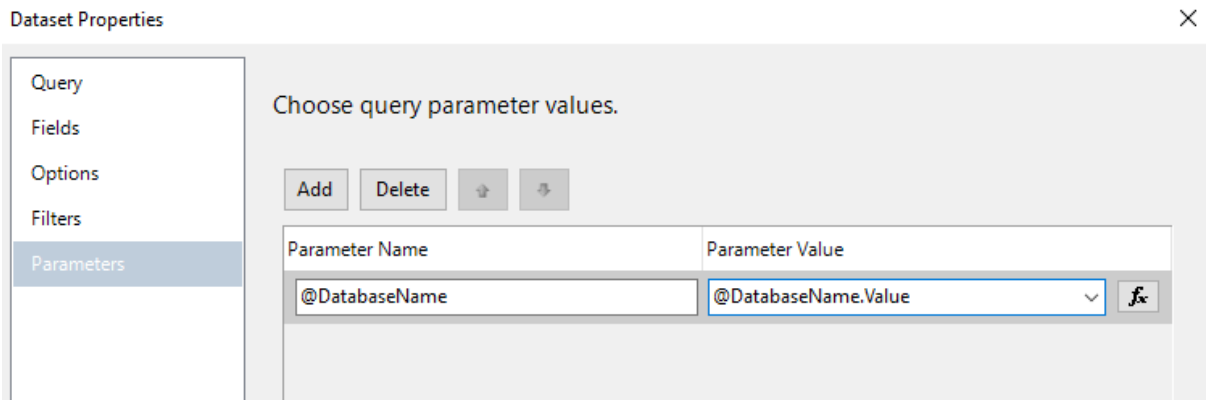

In the Report Data pane, right-click Datasets, and click Add Dataset.



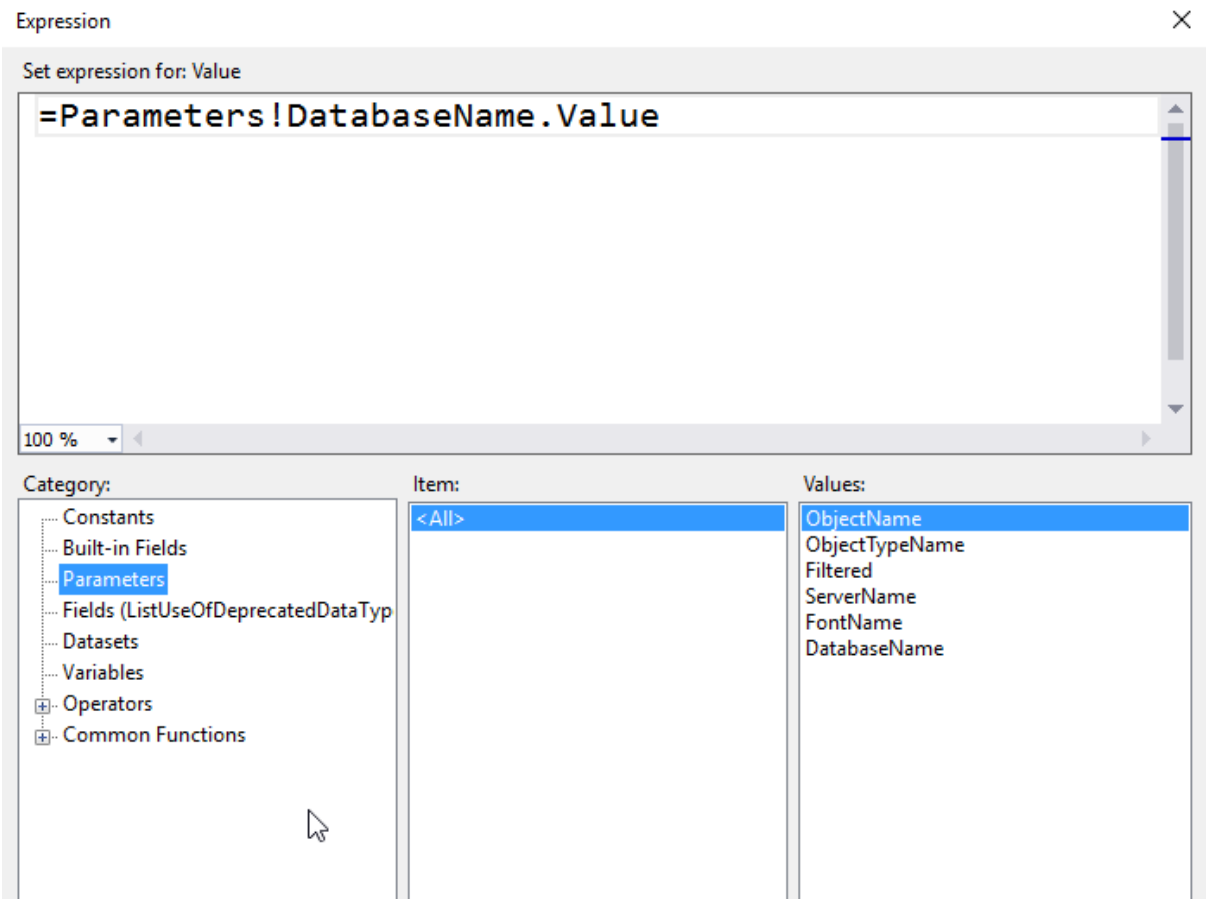

Configure the dataset properties as follows:



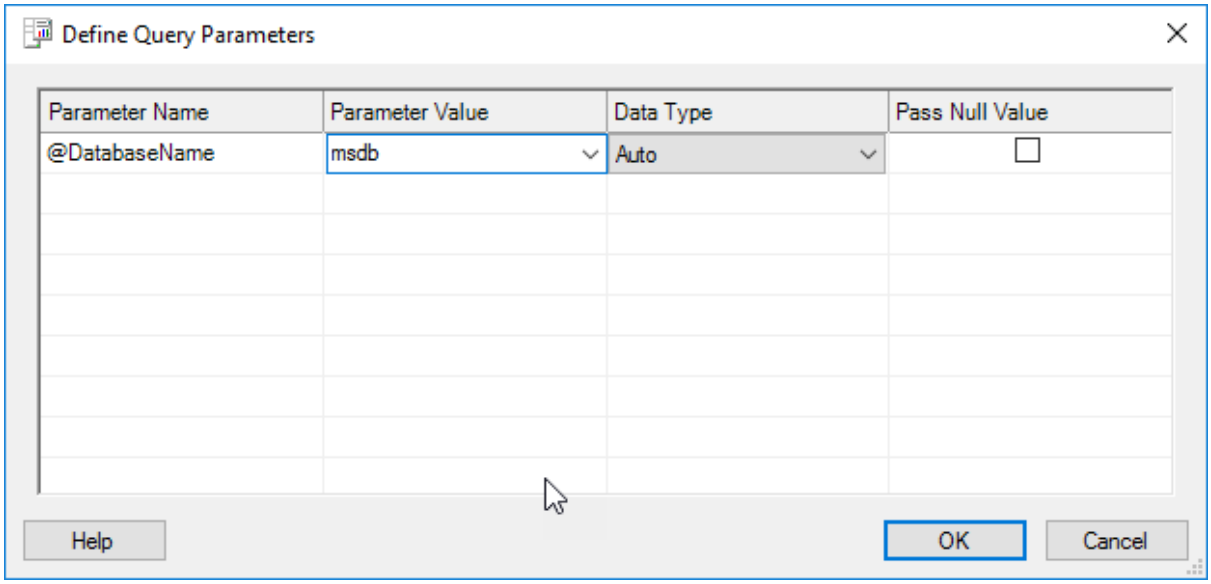

Also, for the dataset, make sure the Parameters collection is set like this:




If you need to change it, use the function icon and do this:

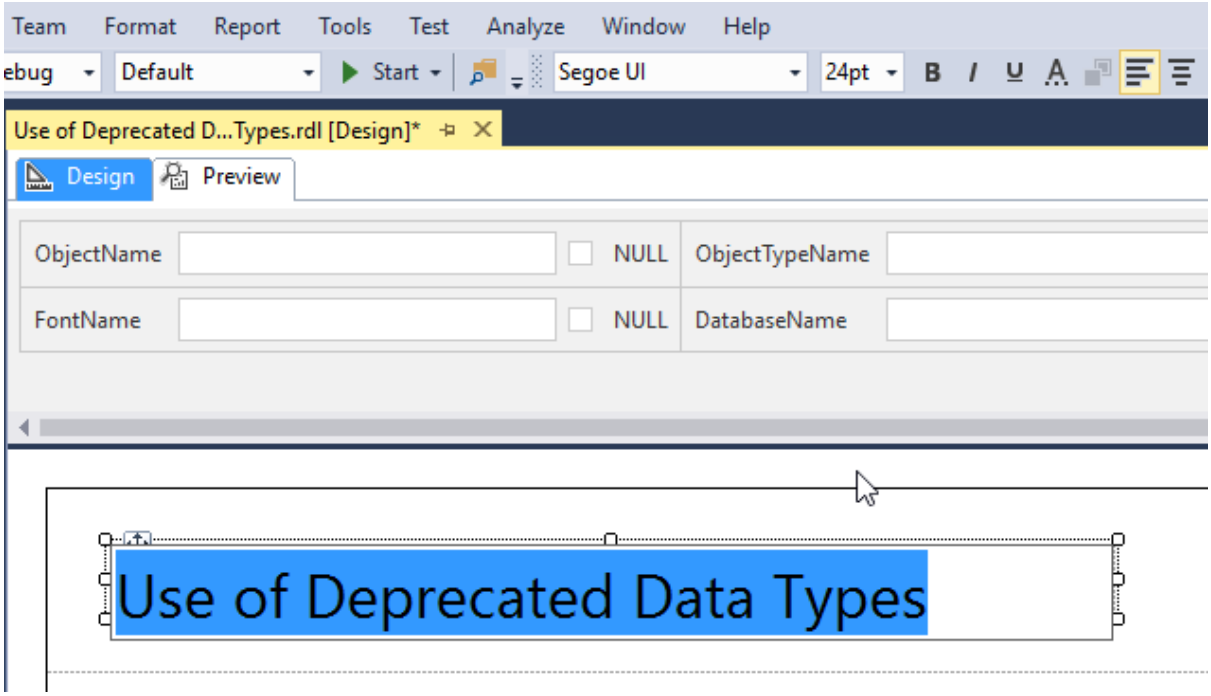



If you try to save it, you'll be asked for a parameter value so that the fields can be refreshed. Type msdb for the database name parameter.

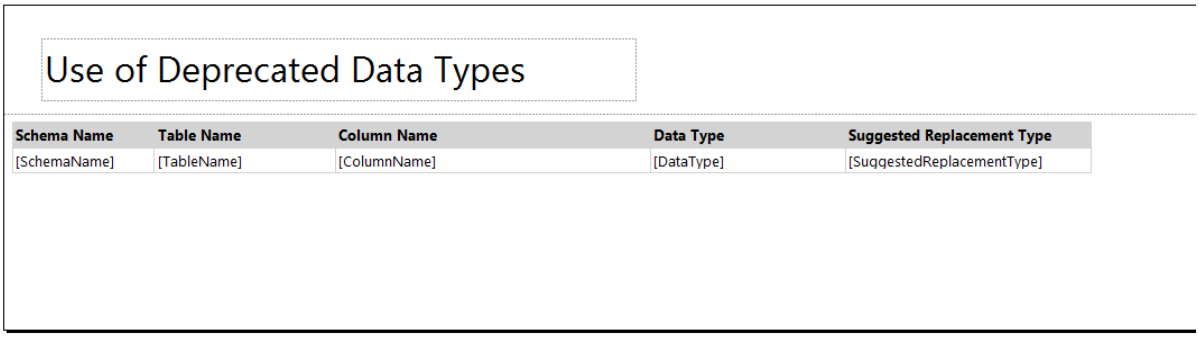



Right-click in the open space of the report and click Add Page Header.

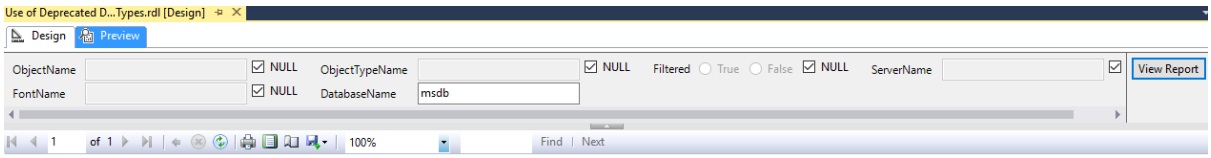

Drag a textbox out from the Toolbox, enter a name, and change its size:

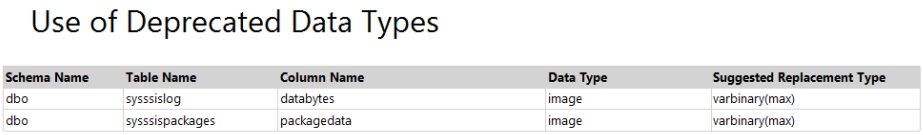


From the Toolbox, drag a table onto the body of the report, then drag the columns out, and resize the table:

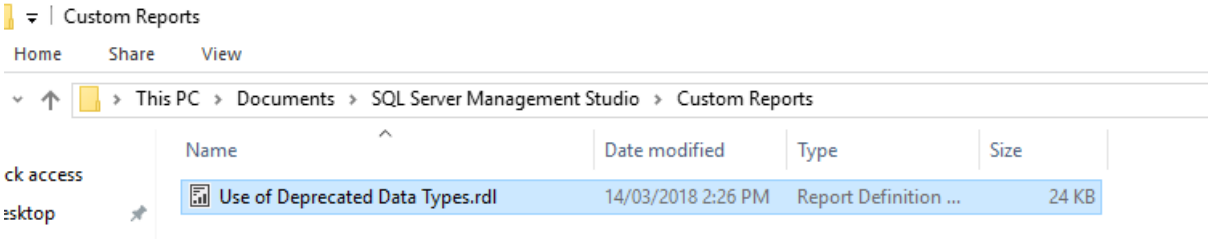



You should now be able to preview the report. Click the Preview tab, enter msdb for the database name, and click View Report.

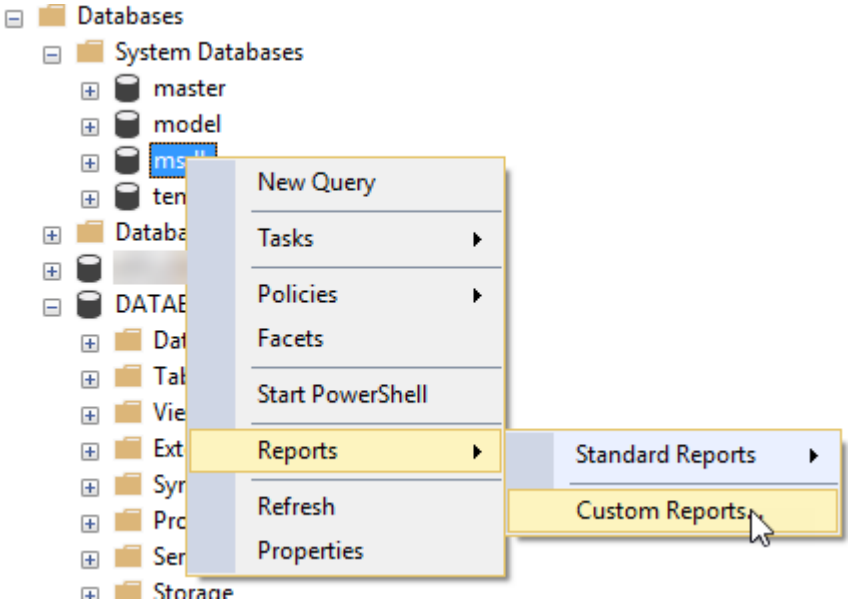


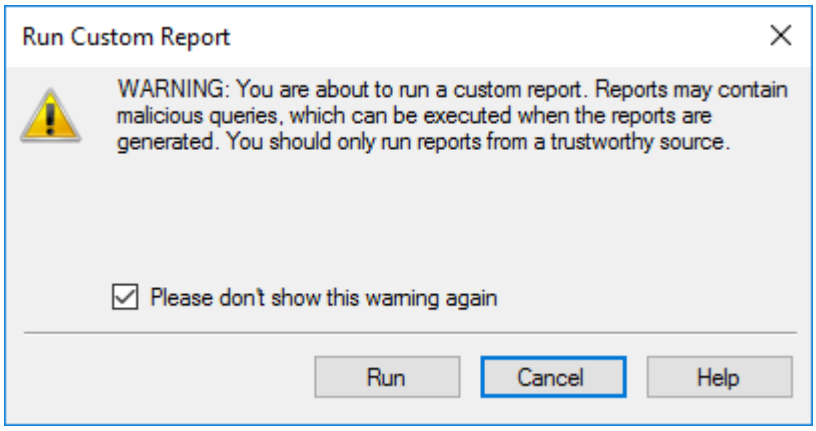

Now we're ready to push the report out. Copy the rdl file to the following folder:



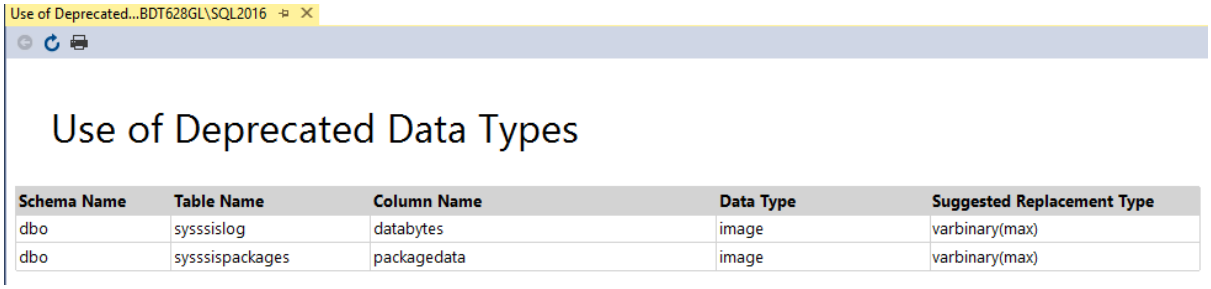

Back in SSMS, right-click the msdb database, then click Reports, then Custom Reports.




You'll get a warning which you can choose to not show again.

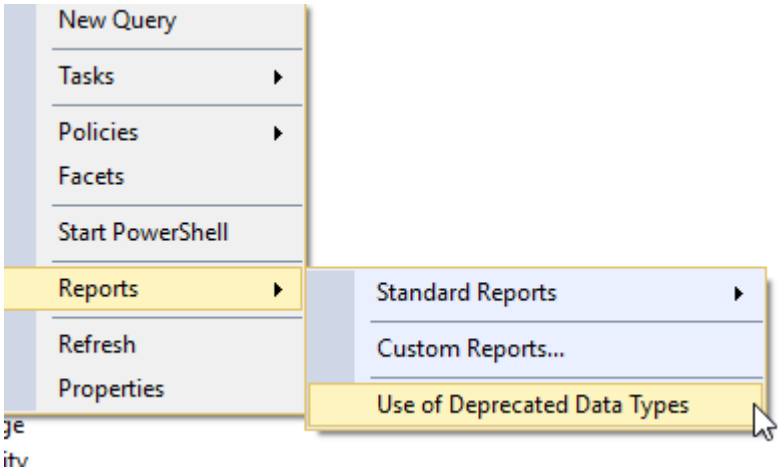



Then your report should appear.




Try it on different databases and you'll see it works well. Note though that we haven't displayed the database name. That should be added to the page header but is left as an exercise for you to try.

Another thing to notice is that if you try to run it again, it appears in the drop-down list, for any object of the same type.




Custom Reports were a useful addition to SQL Server Management Studio.

# 2 Appearance

## 2.1 Environment Fonts

I've been very lucky over the years because I haven't needed to wear glasses. Every now and then I purchase a pair because I thought it might help with reading. Once I get them though, I find them more inconvenient than helpful and stop using them. I've am long-sighted in one eye and short-sighted in the other. That's turned out to be a really useful thing in day to day life.

However, where this comes unstuck is on modern laptops. There seems to be a current trend to pushing more and more pixels into the same size laptop screens, but the applications aren't helping to deal with that.

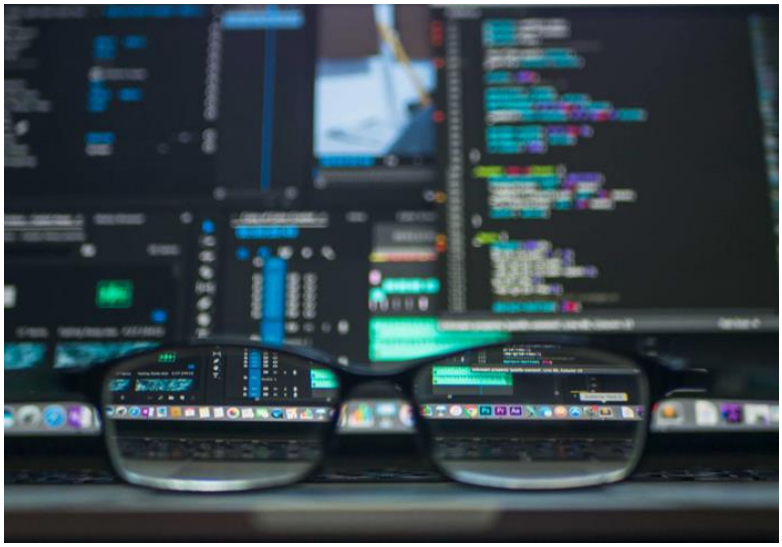


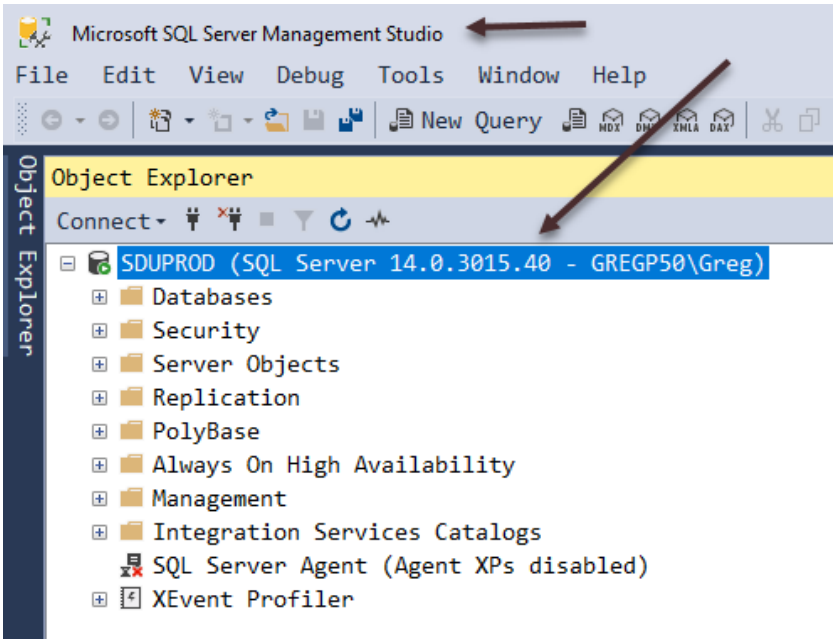
Image by Kevin

I'm sorry, but even a full 1080p screen (1920x1080) is crazy small on a 13-inch laptop, pretty bad on a 14-inch one, and ok on a 15-inch one.

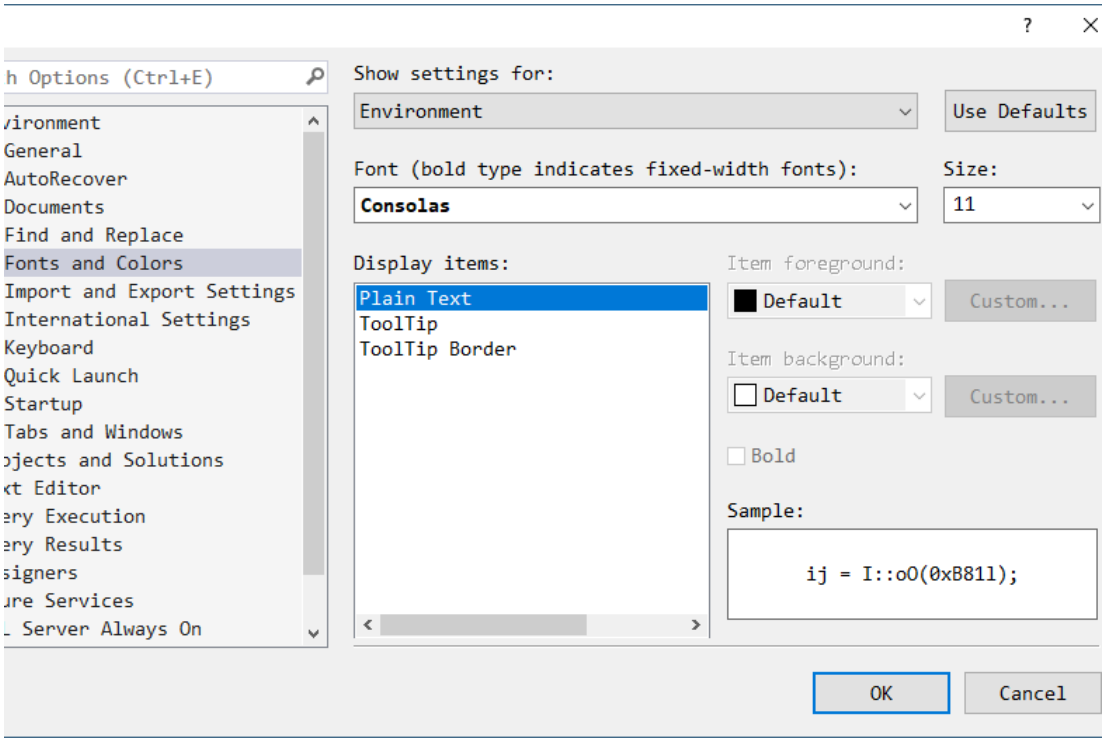
An update to Windows 8 tried to fix things by **automagically** scaling everything. I hated that with a passion. I'd move things from one screen to another and things would radically change size because even though the screens had the same resolution, they had different DPI (dots per inch) settings and Windows decided to auto-scale it for me. So I found the option to kill that.

In Windows 10, **SQL Server Management Studio** is pretty usable for me. I can set fonts or use zoom as needed. The one thing that used to be a problem on small laptops though was the text used in areas like Object Explorer.

For some versions now, you've been able to fix that. Notice that on my screen, the title bar is quite small, but the Object Explorer text is much more readable:




I've found that many people don't realize that in recent versions of SSMS, this can be changed. In Tools>Options>Fonts and Colors, you choose **Environment**:




(Notice that the text in this tool window is also larger and more readable)

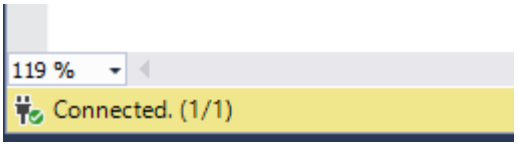
What you'll find though, is that you can't just directly change the size. You need to change the font first. In this case, I chose **Consolas**. Then, you can set the size.

I hope that helps someone else avoid glasses for just a little longer. A bonus is that is makes presentation screens look better to the audience as well.

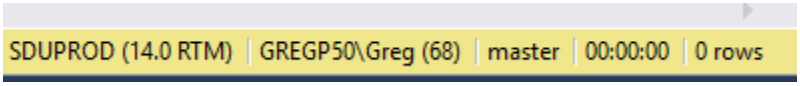
## 2.2 Changing status bar values shown

The status bar at the bottom of a query window in SQL Server Management Studio contains a wealth of information in its default configuration.

The bottom left shows the connection state:



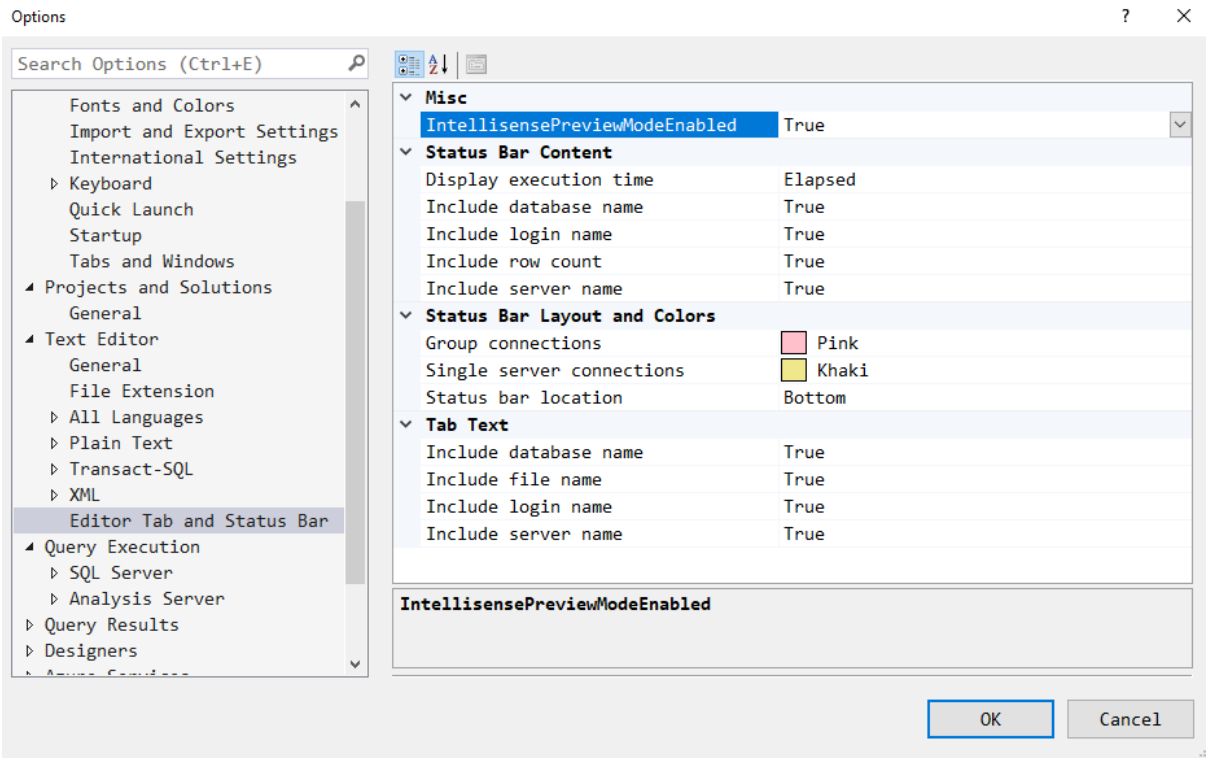

And the bottom right shows quite a bit:




In this case, it's showing me that I'm connected to a server called SDUPROD; it's running v14 of SQL Server (ie: SQL Server 2017); I'm logged on as GREGP50\Greg; my spid (or session ID) is 68; and I'm connected to the master database.

If I had a query running, the time would be counting up, and if a query had completed, it would show me the number of rows.

While this is useful, it's also configurable. The options are in Tools > Options> Text Editor > Editor Tab and Status Bar:



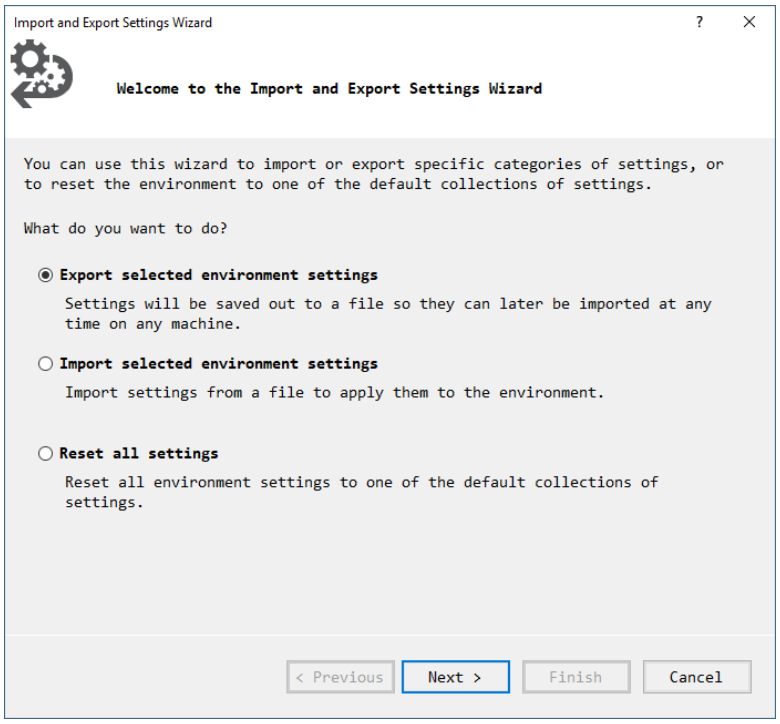

You can choose whether the displayed time is the elapsed time or the end time. You can also remove the display of the database, login, row count, and/or server name. You can set the color used for group and/or single server connections and choose where the status bar is located. You can move it to the top.

## 2.3 Import and Export settings

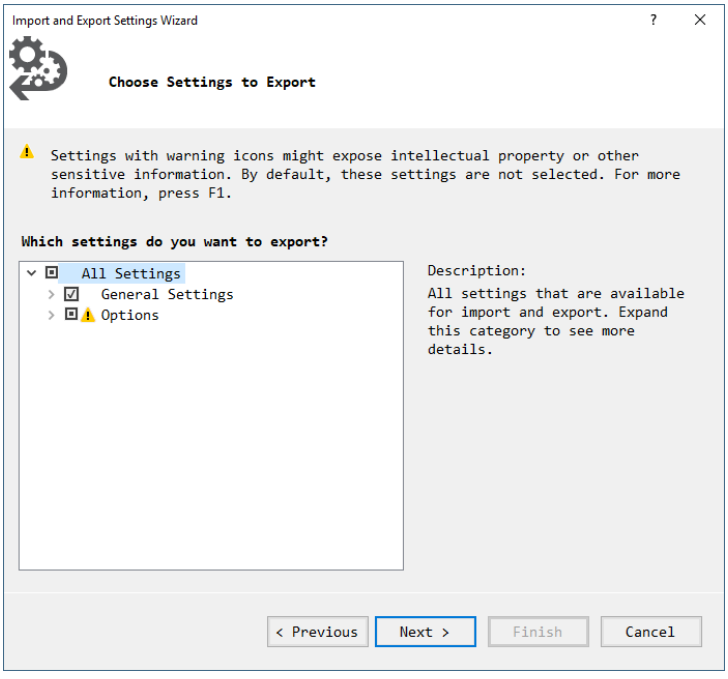
Whenever I need to work on a new laptop or server, or whenever I change versions of SQL Server Management Studio, I kick myself for not remembering to export my settings, so I can import them again.

I spend quite a bit of effort getting SSMS configured the way I want, so it only makes sense to save the settings. Saving them isn't perfect but it's far better than not having done it.

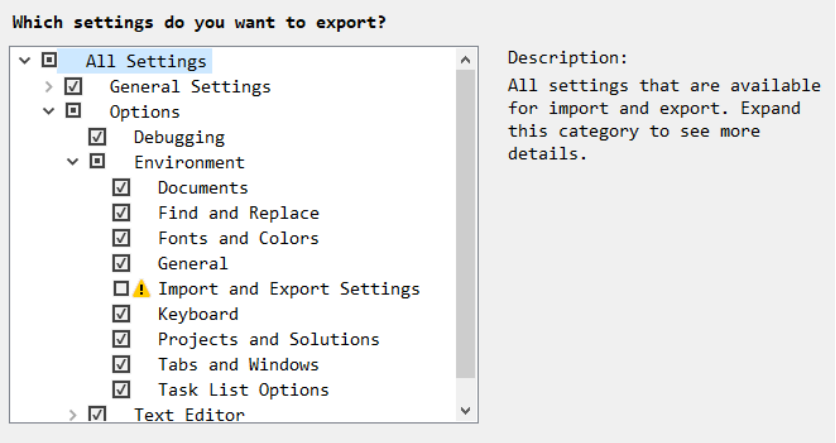
From the Tools menu, choose Import and Export Settings:



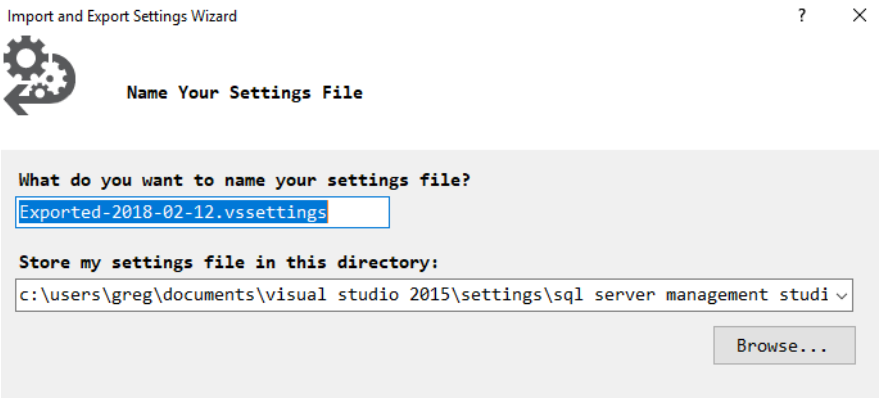

As an example, let's export the settings. I'll choose Next:



Notice that it's not an all or nothing export. I can choose details of which settings or groups of settings to export.




In this case, I wanted all of them, so I just need to pick a name and a location:




And next time I change to a different machine or new version, I can just import them and pat myself on the back for remembering.

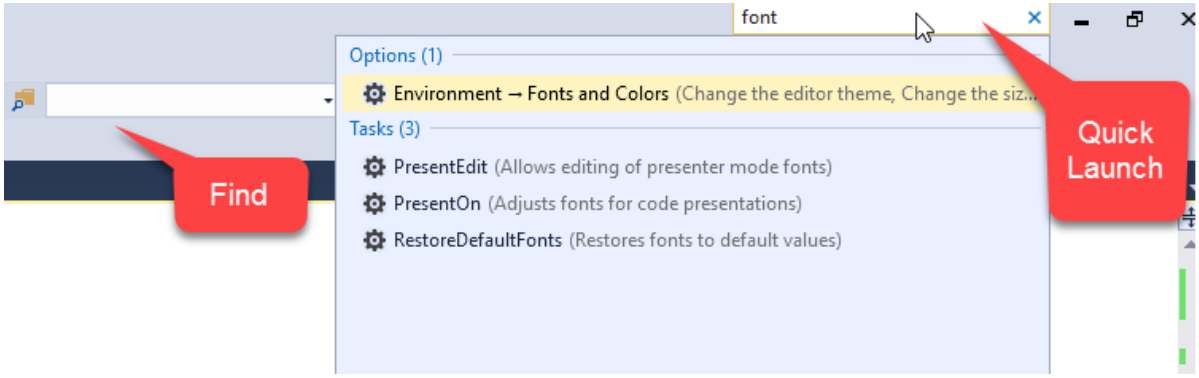
## 2.4 Presentation Mode

I spend a lot of time delivering presentations of various types. Many of those presentations involve showing code in either SQL Server Management Studio (SSMS) or Visual Studio (VS).

I've become quite fast at taking a default setup of SSMS and changing it to the fonts, etc. that I want to use for a presentation. Given how large these fonts are, I don't want to use them for day to day work.

The best solution that I've found for this is to create another user (let's call it DemoUser) on my laptop, and then configuring fonts, etc. for presentations for that user, quite separate to my normal work fonts.

In recent builds of SSMS, the team realized the importance of this and added a Presentation mode. In this image, you can see that I've typed **font** into the Quick Launch bar. (Note that this is separate to the Find box that I've also shown. I've seen people try to type "font" into that box and wonder why nothing relevant came up).




In the drop-down list that appears, you can see a list of options that contain the word **Font**. It checks both the option name, and the description. While you could type **Present** as I've seen many people suggest, you'll note that it wouldn't bring up the option to **RestoreDefaultFonts**. The word **Font** is present in all the options that we want.

Let's start by choosing **PresentEdit** from that list.

qlt_presentermode_defaults.xml

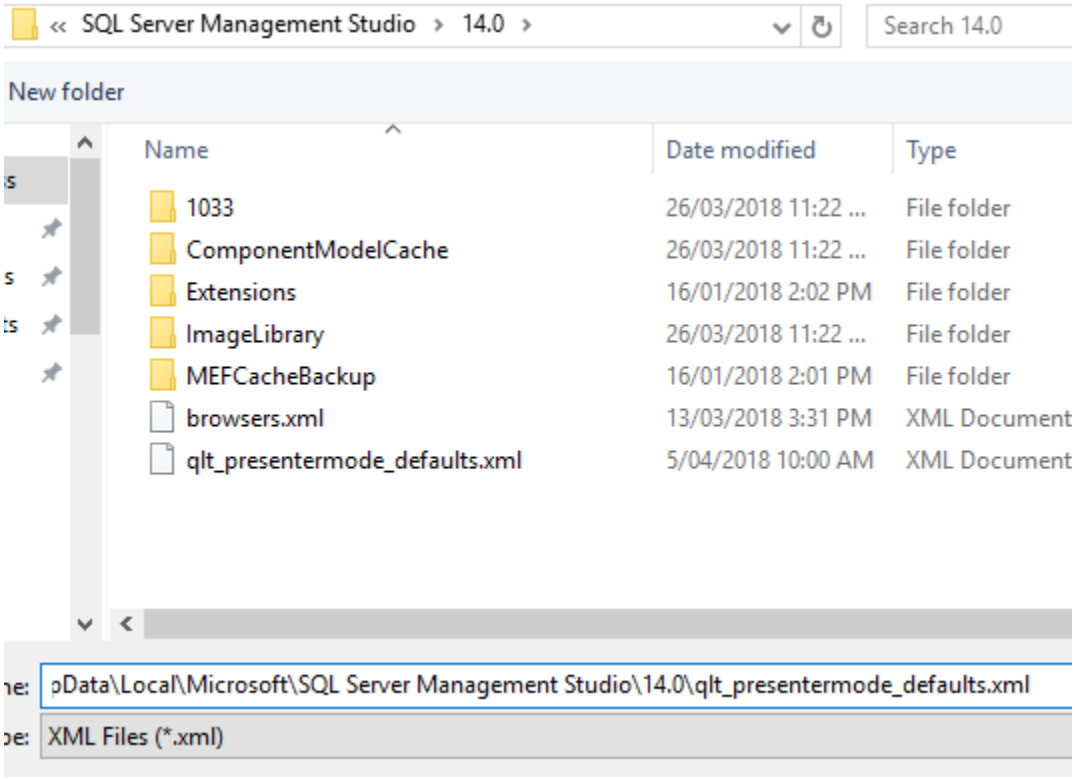
```
<?xml version="1.0"?>
<PresenterModeSettings>
  <EnvironmentFontFamily>Segoe UI</EnvironmentFontFamily>
  <EnvironmentFontSize>11</EnvironmentFontSize>
  <TextEditorFontFamily>Consolas</TextEditorFontFamily>
  <TextEditorFontSize>14</TextEditorFontSize>
</PresenterModeSettings>
```

You'll note that an XML file appears. It would be helpful if this was a GUI instead of an XML file because you'll need to know what the names of the other settings are, if you want to change them.

For now, let's just change the TextEditorFontSize and the EnvironmentFontSize using the perfectly-fine XML editor in SSMS. The first entry changes the size of the text when you're editing queries. The second one affects the size of text in Object Explorer, menus, etc.

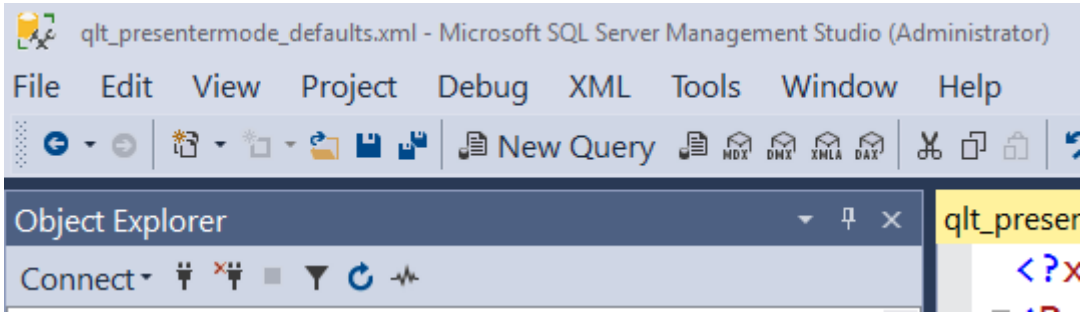
```
<?xml version="1.0"?>
<PresenterModeSettings>
  <EnvironmentFontFamily>Segoe UI</Envir
  <EnvironmentFontSize>12</EnvironmentFo
  <TextEditorFontFamily>Consolas</TextEd
  <TextEditorFontSize>16</TextEditorFont
</PresenterModeSettings>
```

If I click File, then Save As, notice where this is saved:




It's under your **AppData** folder. Once this is saved though, we can test it.

Type **Font** in the Quick Launch bar again, then select the **Present On** option. You'll notice that things have changed. Here is my menu, etc. Note the increase in size.



If you change output grid or text sizes, you'll still need to restart SSMS to see the outcome. The only difference is that you won't get the warning that you normally do.

And once we're finished presenting, we want to go back to the sizes we had before. And this is where I'm not at all happy with this presentation option. The only choice you have is **RestoreDefaultFonts**. Once you choose that, you'll see it wasn't kidding. You're right back to defaults, not to the fonts you had previously been using.

So while this had the potential to be a good feature, I can't give it a pass. It feels ill-conceived. Surely when you click **Present On**, they could save your current settings, and then have a **Present Off** option instead of, or in addition to, the **RestoreDefaultFonts** option.

Once again, I think there aren't enough grumpy old guys in the team.

## 2.5 Screen and Printing Colors

SQL Server Management Studio (SSMS) is a highly configurable tool. One of the areas that's often ignored but which can be quite important is color configuration.

SSMS color codes SQL scripts (and other types of files that it understands) as you type.

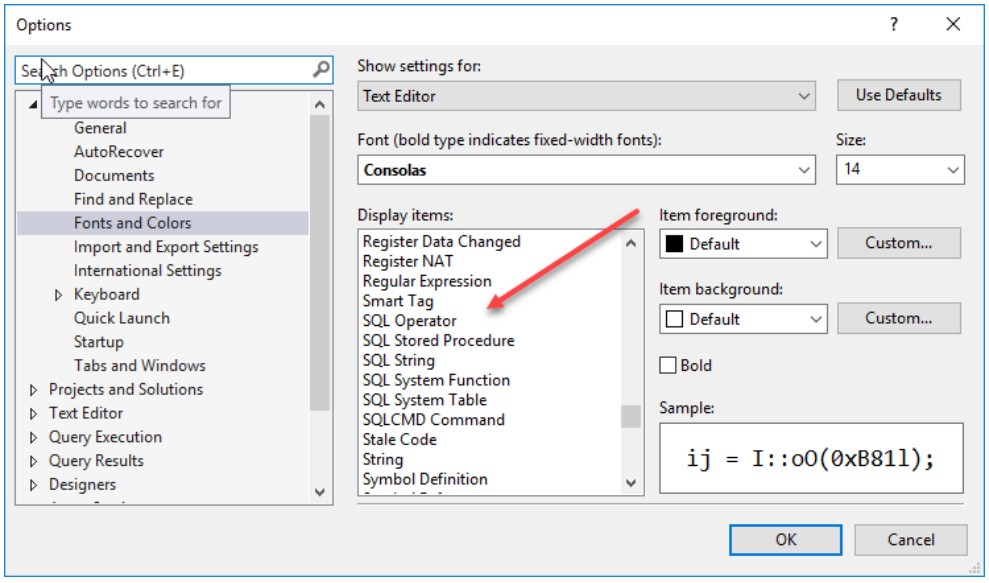
```
/*

SELECT SDU_Tools.CalculateAge('1968-11-20', SYSDATETIME());
SELECT SDU_Tools.CalculateAge('1942-09-16', '2017-12-31');

*/
    RETURN CASE WHEN @CalculationDate >= @StartingDate
                THEN DATEDIFF(year, @StartingDate, @CalculationDate
                     - CASE WHEN DATEADD(day, DATEDIFF(day, @Starti
                                 < DATEADD(year, DATEDIFF(year, @St
                            THEN 1
                            ELSE 0
                        END
           END;
```

This is really useful but I've found on some systems that some of the color selections aren't great. Here's an example:

```
SELECT @@VERSION;
SELECT * FROM sys.tables ;
```

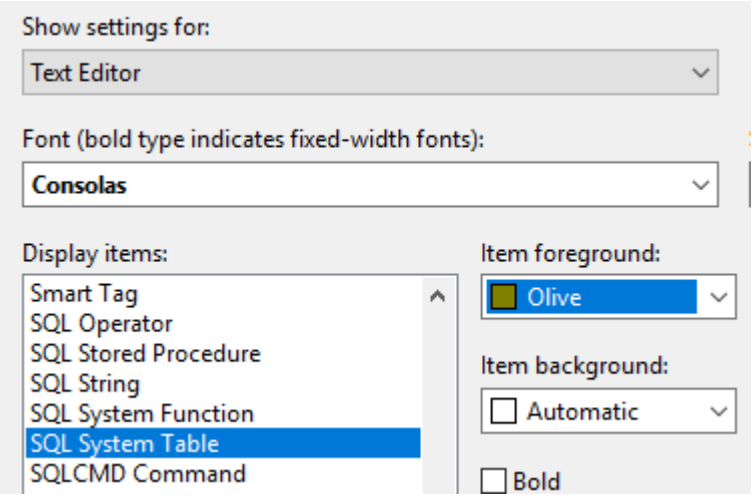
On many systems that I work with, the color for **sys.tables** in the query above is quite a fluoro green and almost unreadable. But if you don't like this, you can change it.

In **Tools**, then **Options**, then **Fonts and Colors**, select **Text Editor**, then look in the list of **Display items**:



Note that when **Text Editor** is selected, several SQL options appear in the **Display items** list. They are not there when you select other settings such as **Printer**.

I could then change the nasty SQL System Table color option to something easier to work with:




I noticed that **Comment** was a standard **Green** so I've chosen **Olive** here, and then on my screen, they look much better:

```
SELECT @@VERSION;
SELECT * FROM sys.tables ;
```

If they aren't dark enough, I could also **Bold** them.

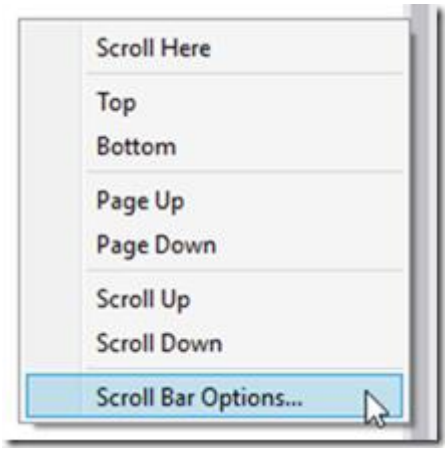
It's worth noting that this can help for people with different visual challenges (or color blindness in general).

While there is a separate set of colors for Printer, up to v17.6 it unfortunately doesn't include the list of SQL language elements. (That does seem odd as it has other language elements).

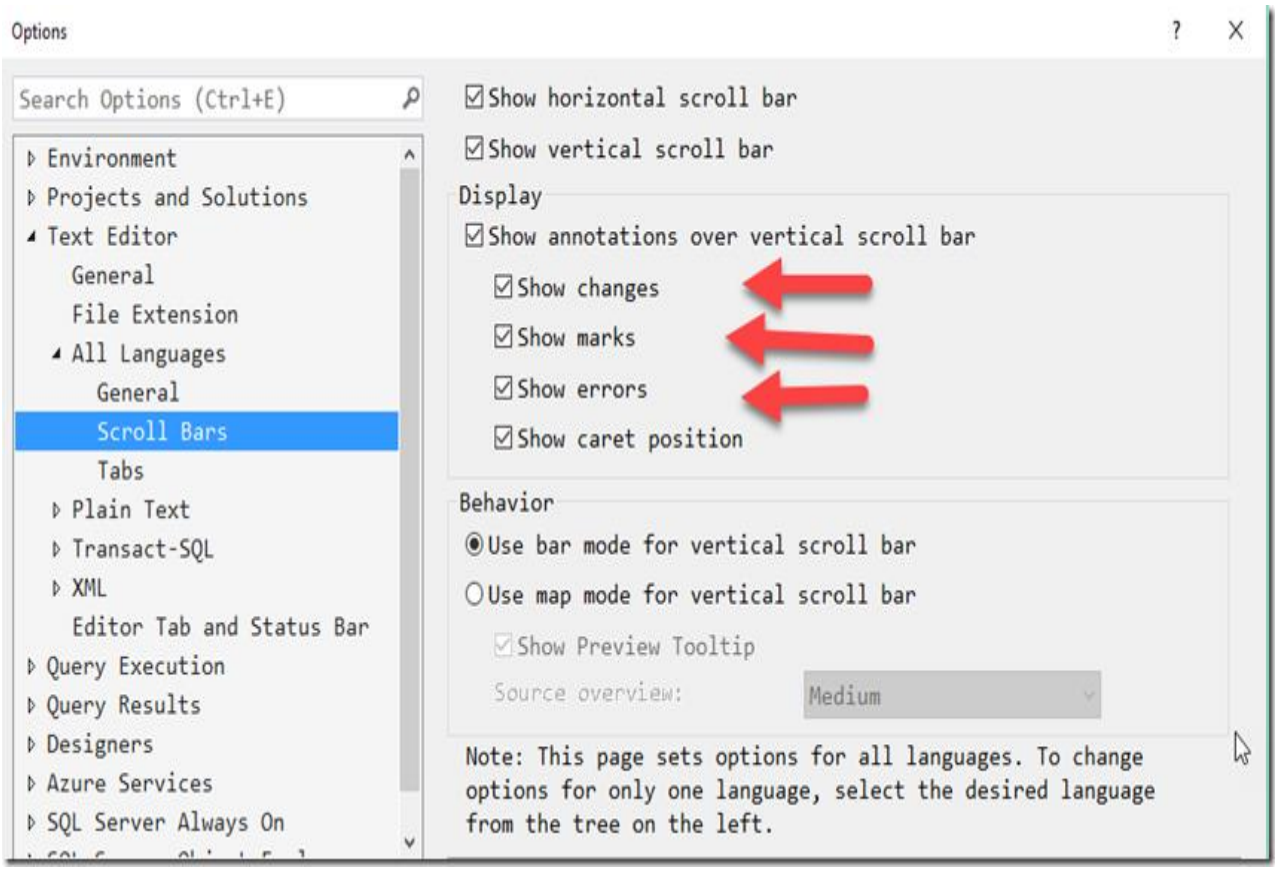
## 2.6 Cleaning up the Scroll Bar

It's great that SQL Server Management Studio has moved into the latest Visual Studio shell. Unfortunately, there are one or two things that are a little harder at first for people who want to use SSMS to write T-SQL. One that was driving me crazy was the scroll bar. Visual Studio tries to give so much information on that bar, about what's changed, where the insertion carat is, etc. The problem with this is that I often now can't even find the handle when I want to scroll the window. For example, how do you grab the handle with your mouse and slide the window when it looks like this?

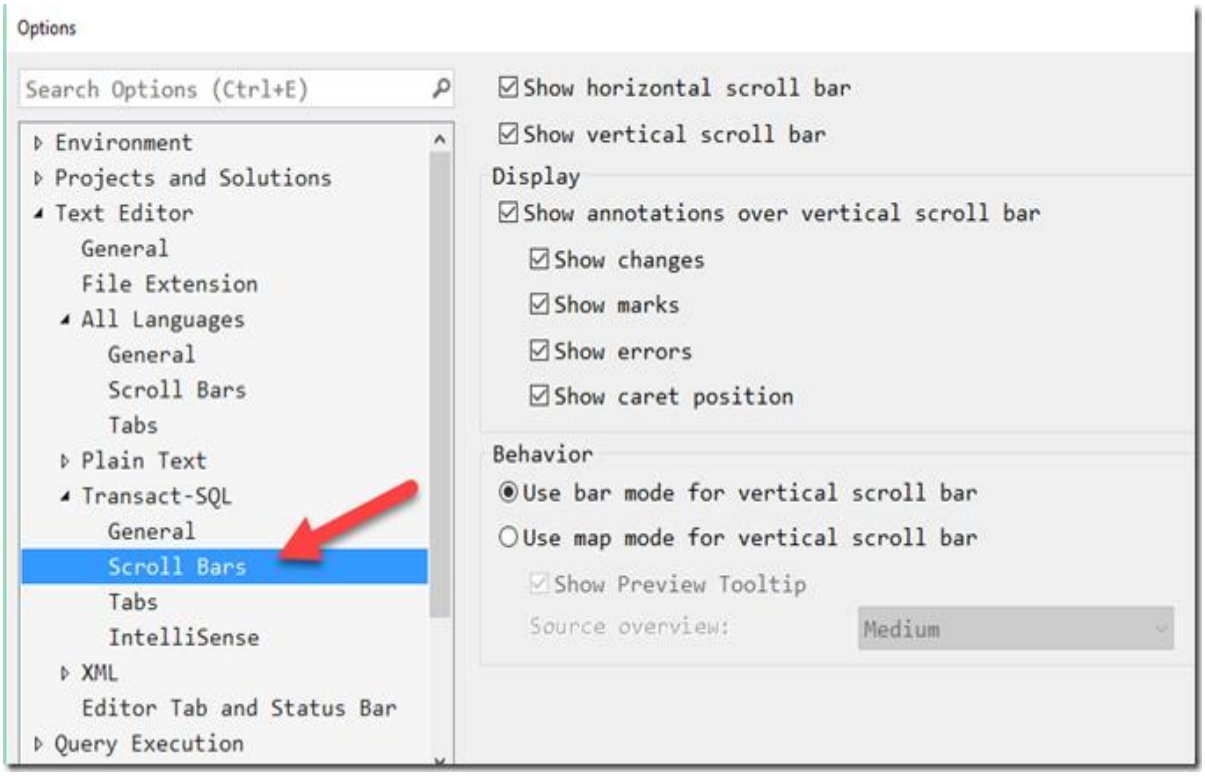

I was starting to get resigned to this when I asked in the MVP email list. Erik Jenson pointed out that the scroll bar itself had properties. I should have thought of that. If you right-click the scroll bar, you get these options:




Choosing “Scroll Bar Options” then leads to this:



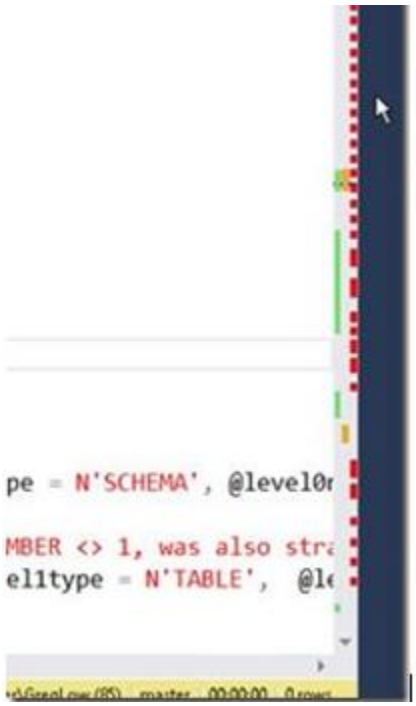

For me, the ones that I’ve highlighted are the real offenders. However, note the warning at the bottom. You really don’t want to remove these for all languages. Some might be helpful to you if you use other languages. So if you do decide to change them, click on the option further down the list, to set them for T-SQL only:




I hope that helps you make SQL Server Management Studio a bit more useful.

## 2.7 Making sense of the colors in the SSMS scroll bar

In an earlier post, I described how I didn't particularly like all the colors that are shown in the scroll bar now in SQL Server Management Studio (SSMS):




In that post, I described how to turn them all off, or at least how to kill off some of them. But, of course they are there for a reason. Instead of turning them all off, you might decide to make sense of what they are there for.

The colors that are displayed are indicating the following:

**Red** - this is showing where syntax errors appear in your code

**Blue** - this shows where the cursor currently is. That's helpful when you have scrolled but haven't moved the cursor. However, given this is the most useful one for me, I have to say that when all the other colors are present, it's the one that I find hard to locate.

**Yellow** - this is indicating changes that you have made but have not yet saved.

**Green** - this is showing saved changes.

**Maroon** - this shows the location of "marks" - for us this means breakpoints

**Black** - this shows bookmarks “Marks” are shown in maroon (Breakpoints) and black (Bookmarks). These occupy the left column

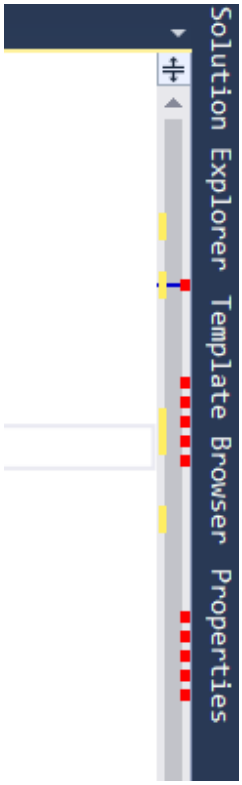
And remember that you can change which ones are displayed by right-clicking the scroll bar and changing its settings.

## 2.8 Scroll bar map mode

In another section, I've described ways to configure the scroll bar in SQL Server Management Studio (SSMS).

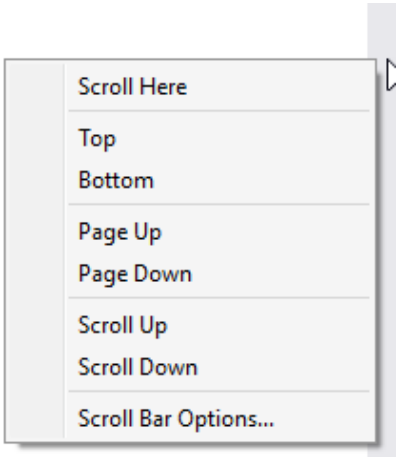
There is another key option that I haven't discussed previously: map mode.

By default, the scroll bar shows the changes, syntax errors, etc:

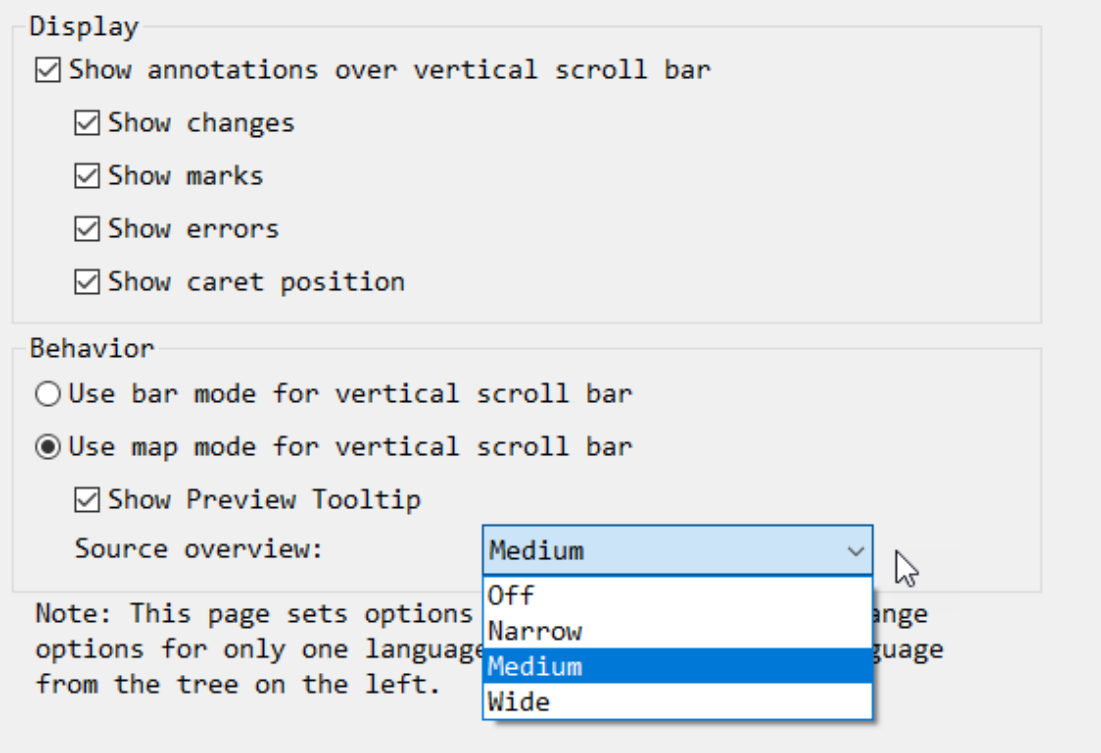



If you have a long script though, it can be hard to visualize what's in the other parts of the script. Map mode can help with this.

Right-click the scroll bar and choose Scroll Bar Options:



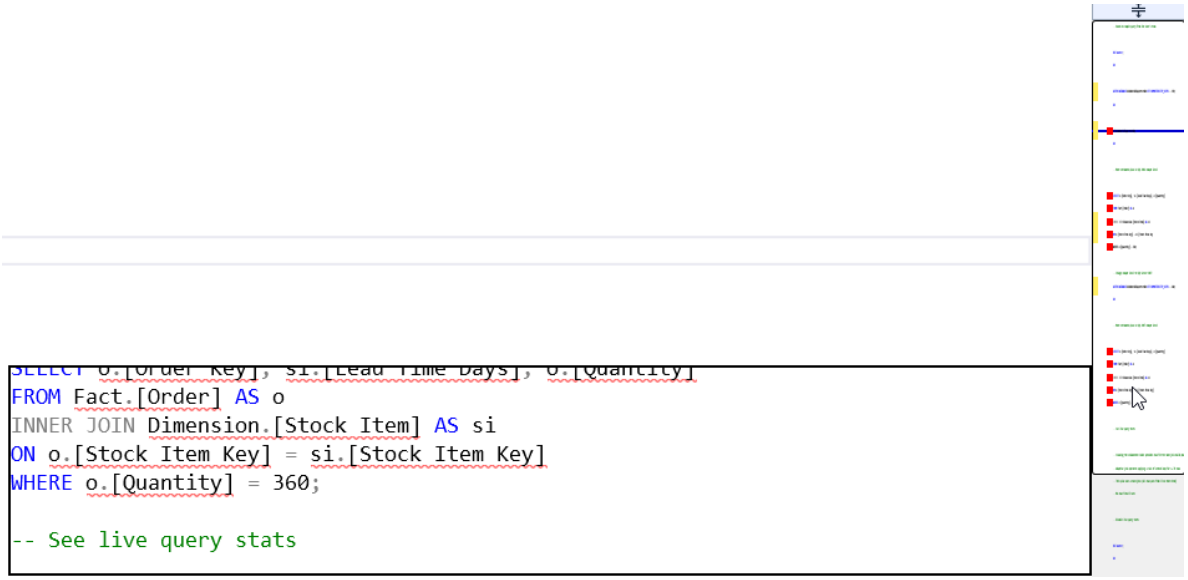

Below the display options that we have previously looked at is another set of options:




When you choose to use map mode instead of bar mode (the default), the appearance of the scroll bar changes:


You are presented with a tiny preview of the script beside the colored indicator. Obviously you can't read that but it can help to visualize the position of the code you are working on within the script.

If you hover over any code section though, it then shows you that code in a preview window:



## 2.9 Adding multi-level undo, redo

Years ago, I had the privilege of presenting "what's new in SQL Server 2012" sessions at many locations around the world. When you present sessions, you can sometimes learn as much as you teach. What I learned while delivering those sessions is that the product group sees what's important in a release very differently to what the attendees do.

Each time there's a new version of SQL Server, there will normally be three or four marketing pillars (groups of functionality), and each will have about eight to ten bullet points.

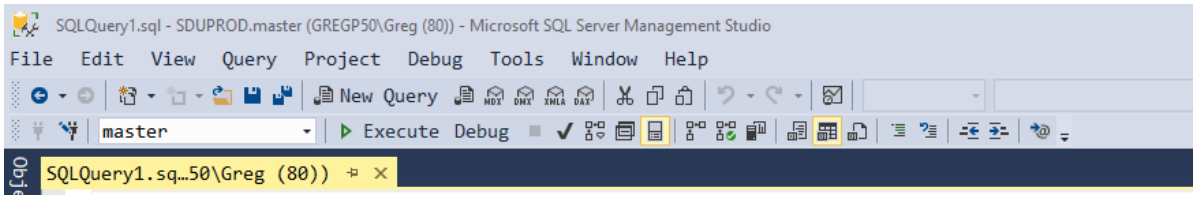
So, when SQL Server 2012 was released, what got the biggest reaction from the audiences? Was it the new availability groups? Was it the new tabular data model in SSAS? Was it the new project model in SSIS?

**It might (or might not) surprise you to hear that the biggest reaction world-wide to the release came when I showed the audience that the undo/redo options in the SSIS designer now actually worked.**

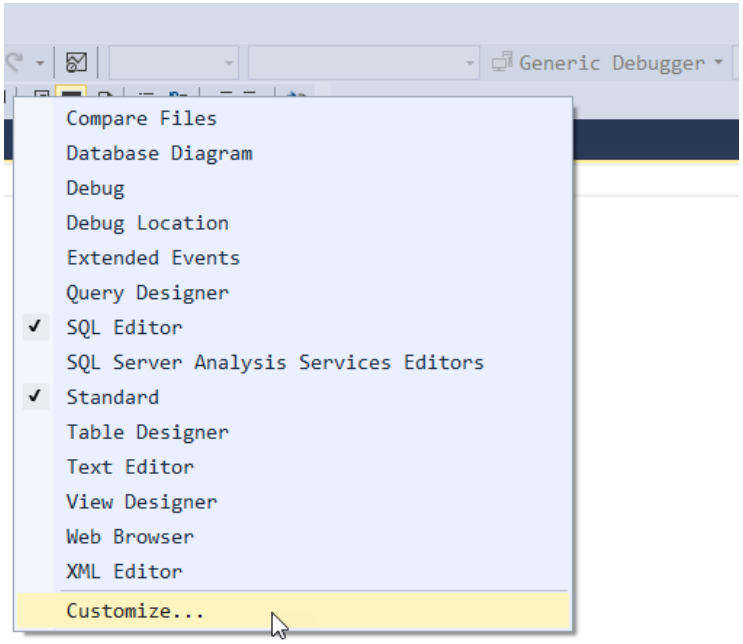
Even better, they were multi-level undo and redo.

SQL Server Management Studio can also use the same option. By default, it only has a single-level undo and redo. It can go back quite a distance in undo but it does so one step at a time. Adding a multi-level undo and redo can really improve your editing experience. I have no idea why it's not there on the toolbar by default. So let's fix that !

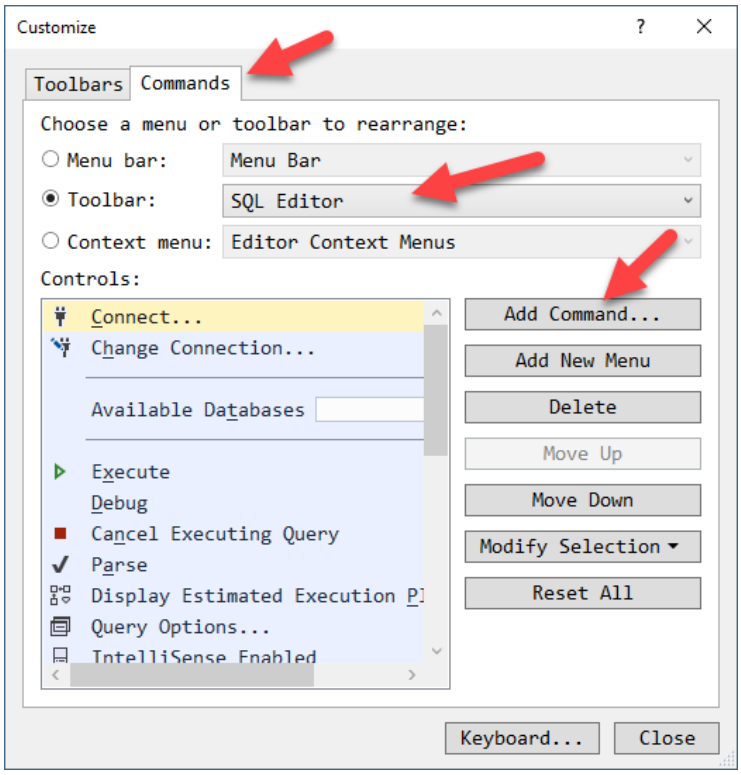
First let's see the toolbar as it started.



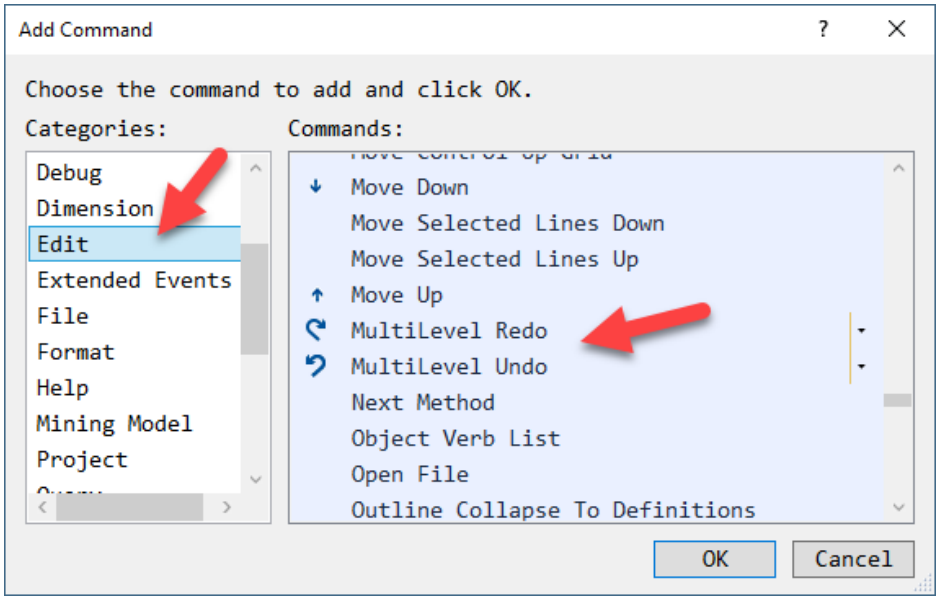

Right-click a blank area in the tool bar then choose Customize.



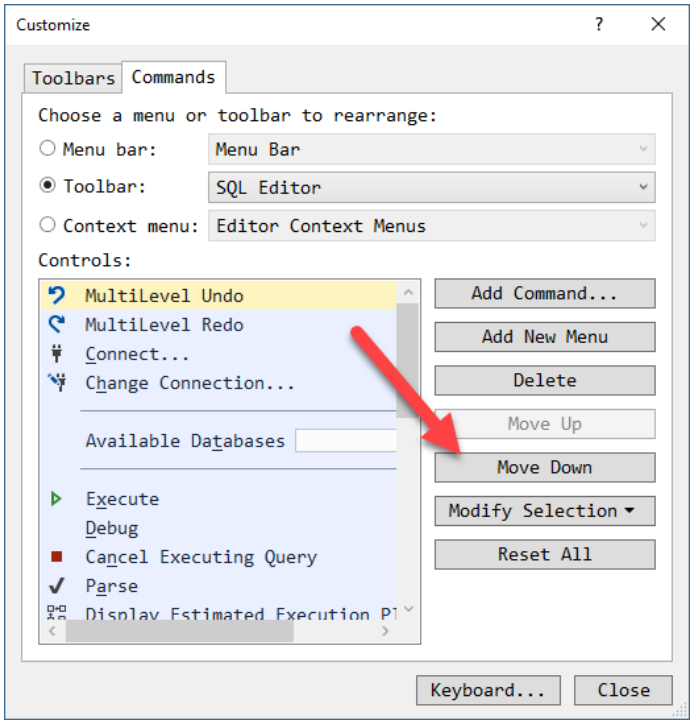
Then choose the Commands tab, the SQL Editor for the Toolbar, and click Add Command:



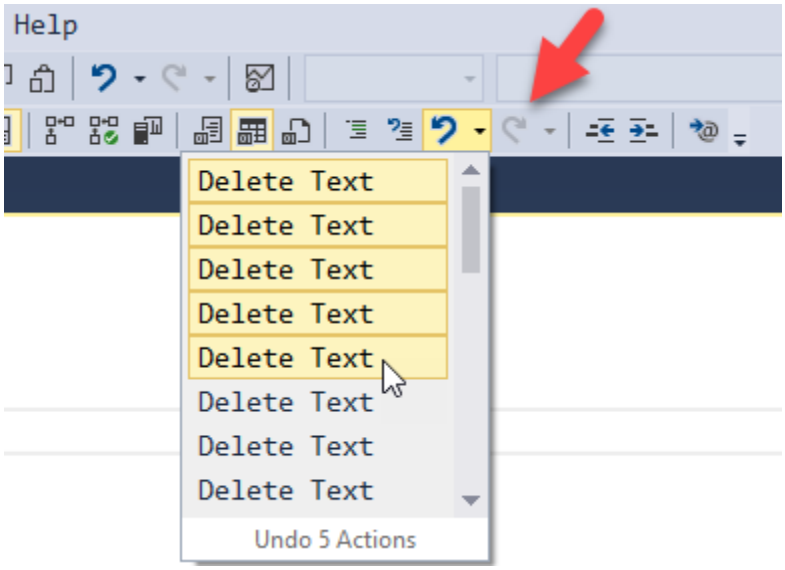

From the Categories, choose Edit. When the commands appear, scroll down to the MultiLevel Redo and click OK.



Do the same to add MultiLevel Undo, then move them down to where you want them using the Move Down button. I like to have them just after the comment / uncomment buttons.




Once they are in place, note that unlike the normal undo (or Control Z), you can click the button, see a list of your previous actions, and choose how many to apply.

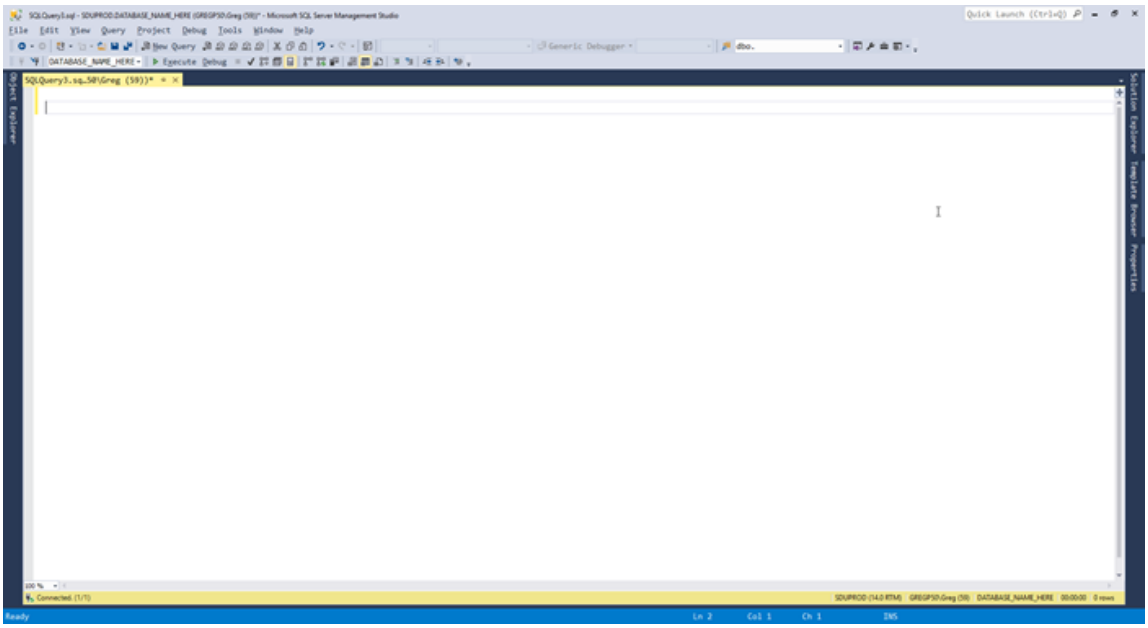



Once you undo some, the same applies to Redo.

## 2.10 Toggle full screen (Alt shift enter)

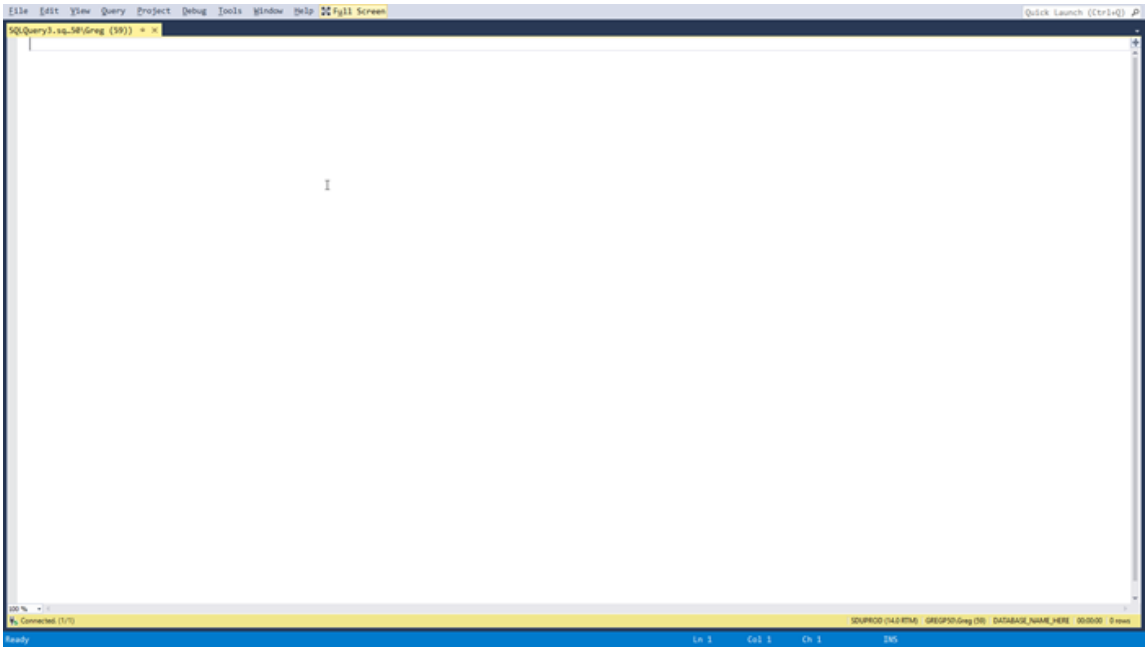
SQL Server Management Studio (SSMS) is a great tool and it has lots of helpful menu items and toolbar items. Unfortunately, all these items take up screen real estate.

You can see that the default screen layout could be considered a bit cluttered if you really just want to focus on the particular query that you're working on.



A keyboard shortcut can help here. **Alt-Shift-Enter** toggles full screen mode in SSMS.

Note how it gives you much more screen real estate to work with:



And the same shortcut toggles it back.

## 2.11 Use visual glyphs for word wrap

Code quality has always been an important topic ever since coding began. Code complexity is an important part of this. One of the topics that came up many years ago was a discussion on what length procedures or functions should be, before they became too difficult to follow.

**I remember one guy commenting that he thought as soon as all the code didn't fit on your screen any more, you were much more likely to have bugs in it.** At the time, screens weren't all that big.

Even today when writing T-SQL though, if you've ever worked on a 4000-line stored procedure, you'll know how hard that is. But if you must work on a long script, it makes it even harder if it's also a wide script. Then you end up scrolling around in both directions. For this reason, most formatting tools like SQL Prompt have an option to limit the width of lines.

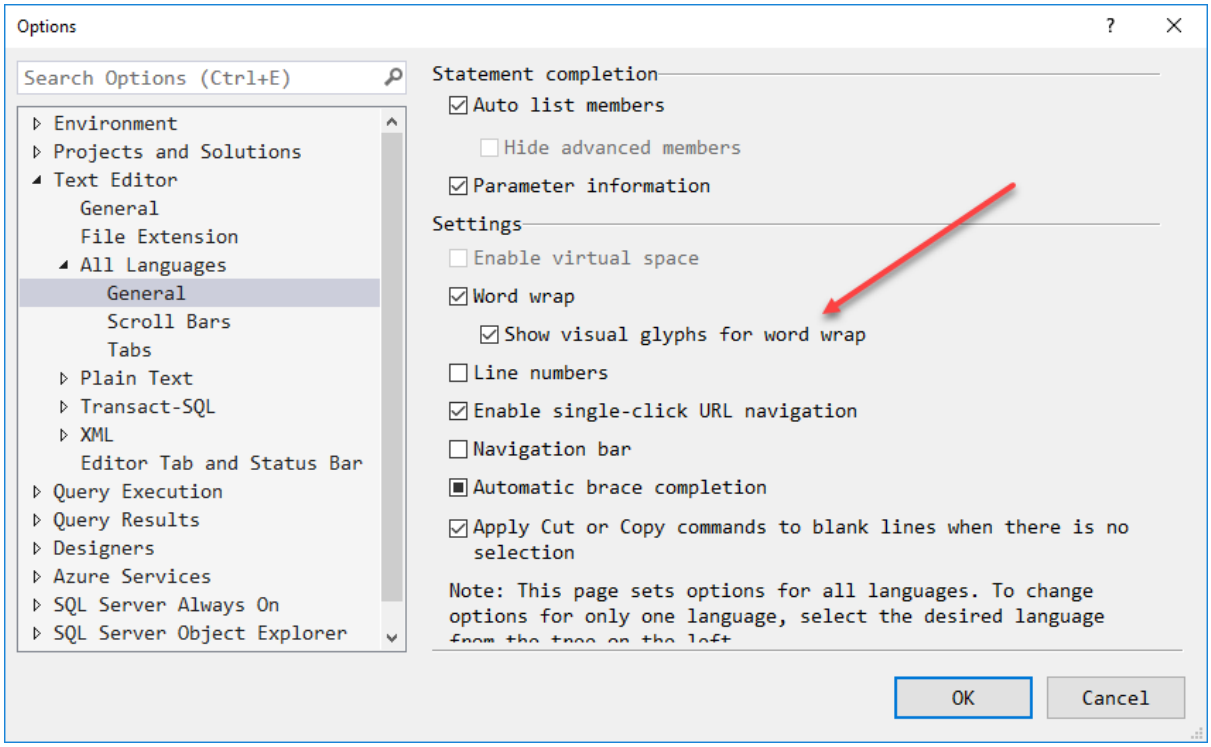
Breaking code into separate lines however, isn't always possible. And if you've decided that you really don't want to do that, you can enable word wrap in SQL Server Management Studio (SSMS).

```
AS
BEGIN
    SET XACT_ABORT ON;
    SET NOCOUNT ON;

    DECLARE @TimePeriod datetime = CAST(SYSDATETIME() AS date);

    WHILE DAY(@TimePeriod) = DAY(SYSDATETIME())
    BEGIN
        INSERT Dimension.[Time Period]
        (
            [TimePeriodKey], [Hour], [AM PM Hour Label], [24 Hour Label], [Minute], [Minute Label],

            [Is AM], [Is PM], [AM PM Label],
            [Time Period Label], [Time Period 24 Hour Label], [Time Period Minute of Day]
        )
        SELECT tpdc.[TimePeriodKey], tpdc.[Hour], tpdc.[AM PM Hour Label], tpdc.[24 Hour Label],
          tpdc.[Minute], tpdc.[Minute Label],
               tpdc.[Is AM], tpdc.[Is PM], tpdc.[AM PM Label],
               tpdc.[Time Period Label], tpdc.[Time Period 24 Hour Label], tpdc.[Time Period Minute
                  of Day]
        FROM Integration.TimePeriodDimensionColumns(@TimePeriod) AS tpdc;

        SET @TimePeriod = DATEADD(minute, 15, @TimePeriod);
    END;
END;
```

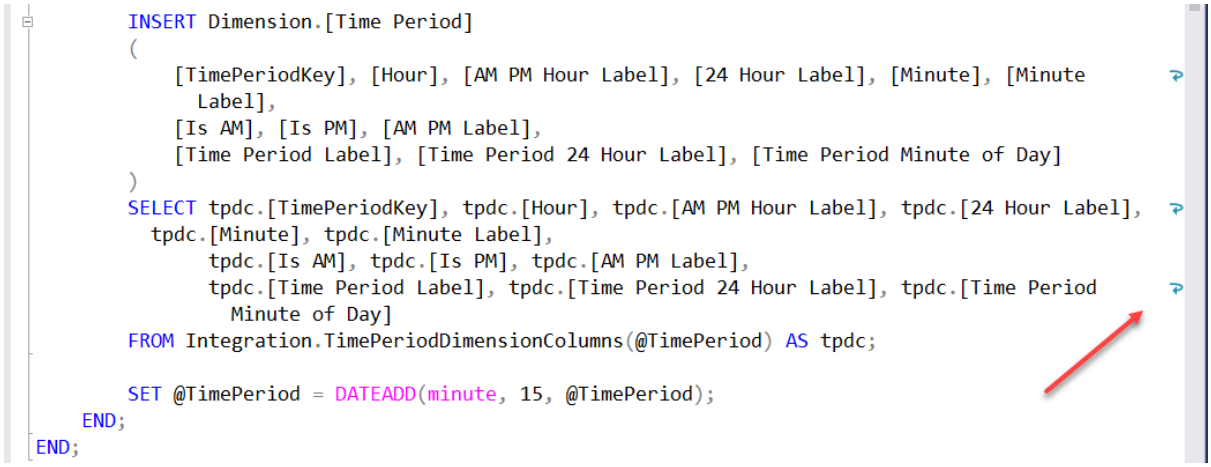
One challenge with this is that it's hard to work out at a glance, which lines have been wrapped and which haven't.

The answer to this is to enable a visual glyph for the word wrap.

From the Tools menu, choose Options, then Text Editor, then All Languages, then General. Note the option for showing the glyphs:




Now when the word wrap occurs, you can see it quite clearly:



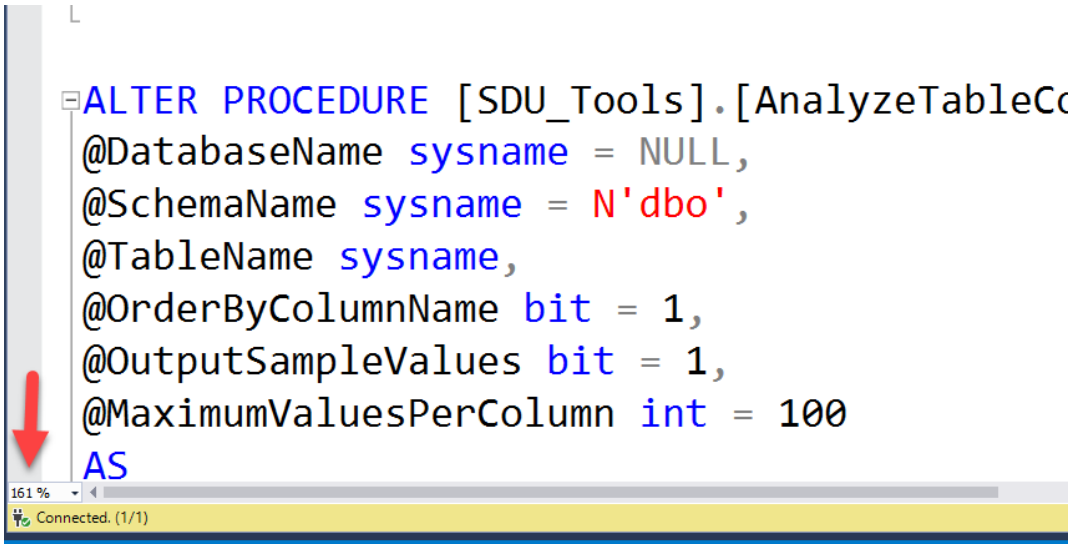
## 2.12 Using zoom features

When working with SQL Server Management Studio, sometimes you need to show someone else what you're working on, and the fonts that you're using are just too small for someone looking over your shoulder or looking at a screen that you've shared with them.

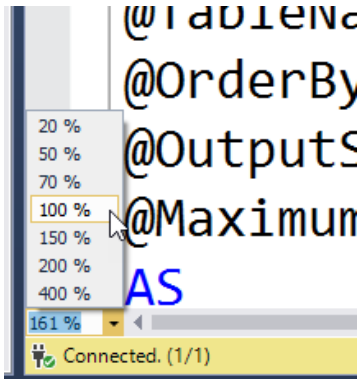
What I often see someone do then, is to go into Tools and Options and start to change the font and color settings. The pain with this is that you then need to set them back later.

There are several options to allow you to zoom in, without needing to change the settings.

First, like many Windows programs, you can hold down the Control key and use the mouse scroll bar to increase or decrease the font size. Note that as you do that, the percentage of zoom is also displayed in the lower right-hand side of the query window.




You could also, of course, just use that drop-down to set the zoom level too:

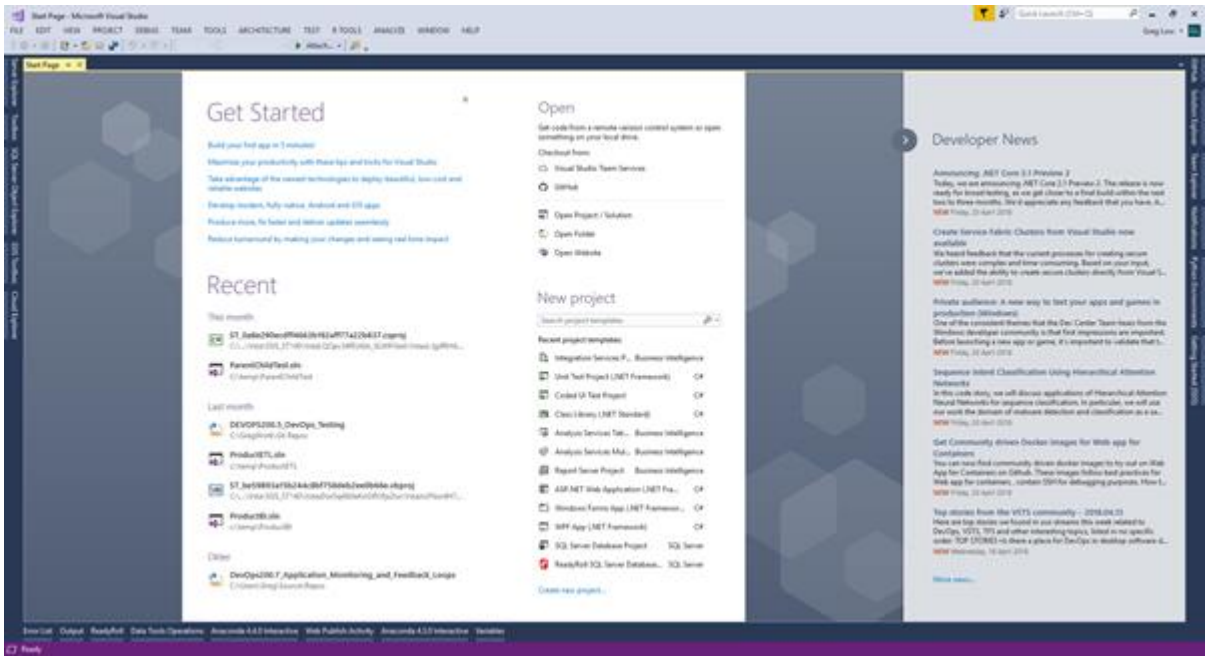



When you make this change, it applies to all current query windows, and all other query windows that you open until you change it back.

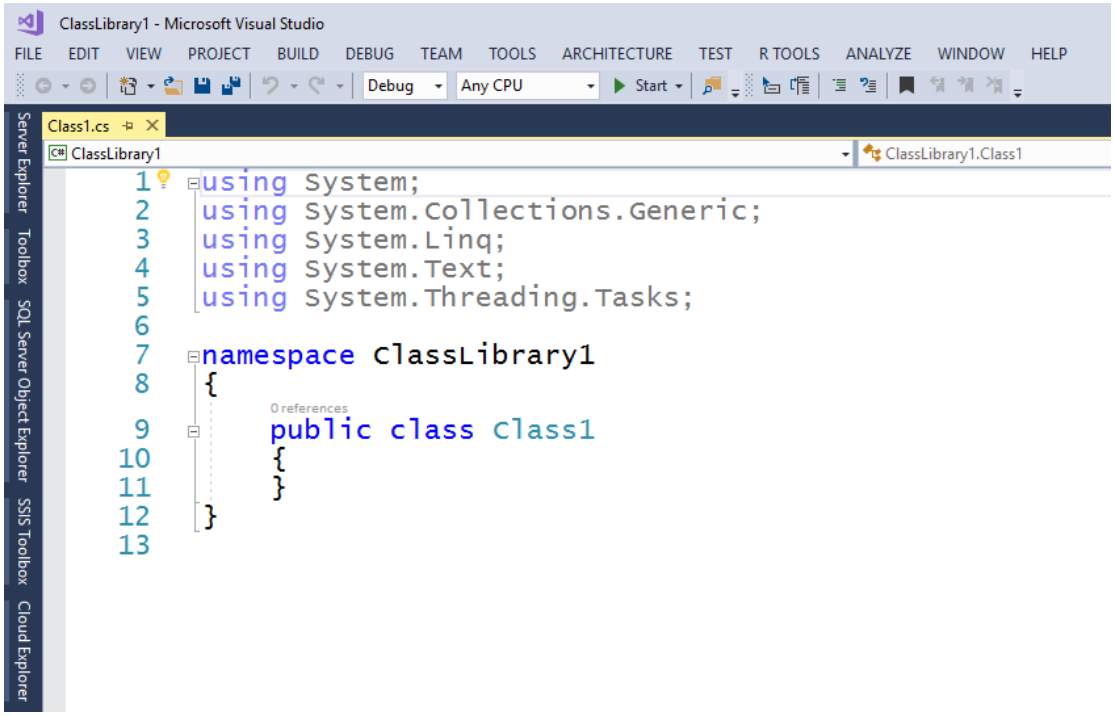
## 2.13 Color theme change

SQL Server Management Studio (SSMS) is based on the Visual Studio shell. What is puzzling sometimes is that features that are present in Visual Studio are not present in SSMS. One of these relates to themes.

"Out of the box", Visual Studio looks like this:

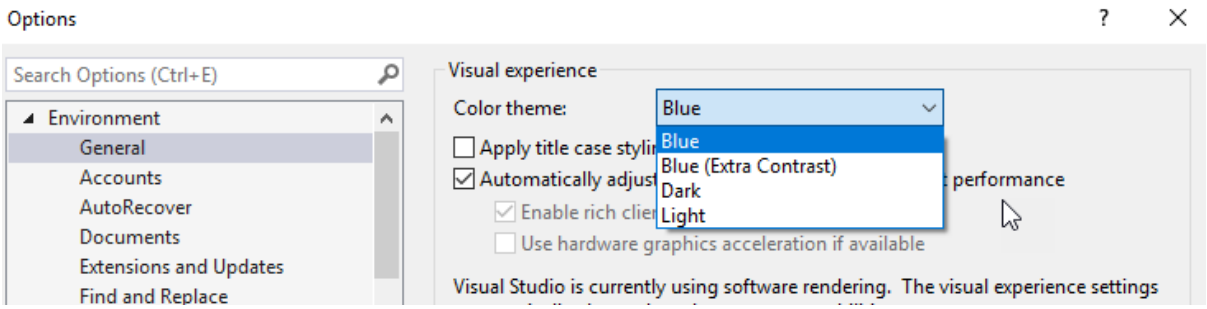



When you write code, the windows look like this:

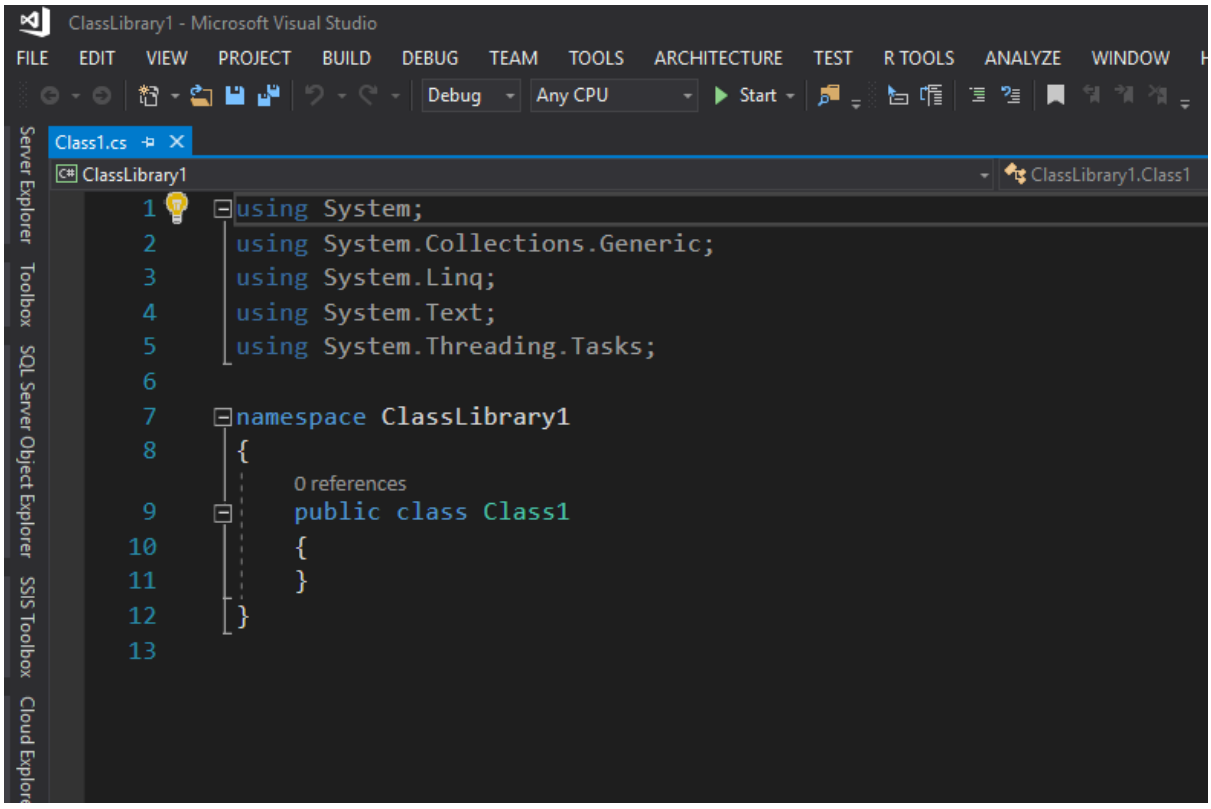


Some people don't like the "whiteness" of the whole screen and find darker colors easier to work with for long periods.

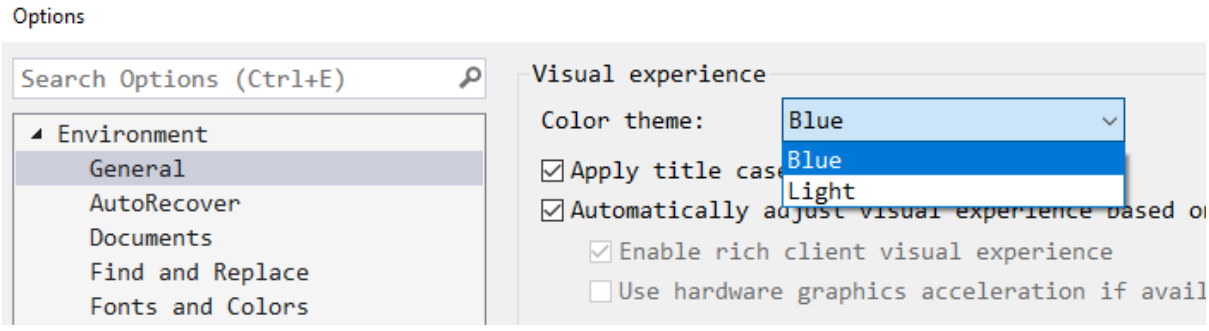
In Visual Studio, you can change this. In Tools, then Options, you can change the theme:




The default here was the Blue theme. There is an extra contrast theme, and a Light theme (can't say I love that one), and a Dark theme. Note that once I choose the Dark theme, I see this:




OK, so what about SSMS? When I go into the same menu items, I see this:



All there is to choose from is the Blue theme that I was using, and the Light theme that I really don't like. It's curious that even the higher contrast Blue theme isn't there.

So what's going on?

Well the team has gone out of the their way to remove the other options. A quick search around the configuration files in the folder:

**C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Binn\ManagementStudio**

leads you to two interesting files: **ssms.pkgdef** and **ssms.pkgundef**.

Opening **ssms.pkgdef** in NotePad++ I can see this:

ssms.pkgdef

```
[$Initialization$]
"ApplicationExtensionsFolder"  = "$RootFolder$\Extensions"
"PkgDefSearchPath"             = "$ShellFolder$\Common7\IDE\ShellExtension
"RegistryRoot"                 = "Software\Microsoft\SQL Server Management
"ImageManifestSearchPath"      = "$ShellFolder$\Common7\IDE\Extensions;\
                                  $ShellFolder$\Common7\IDE\CommonExtensio
                                  $AppDataLocalFolder$\Extensions;\
                                  $ApplicationExtensionsFolder$"
[$RootKey$\Help]
"Product"                      = "SQLServer"
"Version"                      = "140"
"DefaultQuery"                 = "method=f1&query=sql14.portal.f1%00sql14.
"BrandingPack"                 = "SQLServerHelpBranding.mshc"

[$RootKey$\BindingPaths\{687b26ef-c096-4f2d-9f8c-aaafada321ac}]
"$ProgramFiles$\Microsoft SQL Server\140\DAC\Bin"=""

// AutoLoad VS Telemetry package
[$RootKey$\AutoLoadPackages\{adfc4e64-0397-11d1-9f4e-00a0c911004f}]
"{54befe64-0558-4d8c-9fd5-ab0b54733b08}"=dword:00000000
[$RootKey$\AutoLoadPackages\{f1536ef8-92ec-443c-9ed7-fdadf150da82}]
"{54befe64-0558-4d8c-9fd5-ab0b54733b08}"=dword:00000000

```

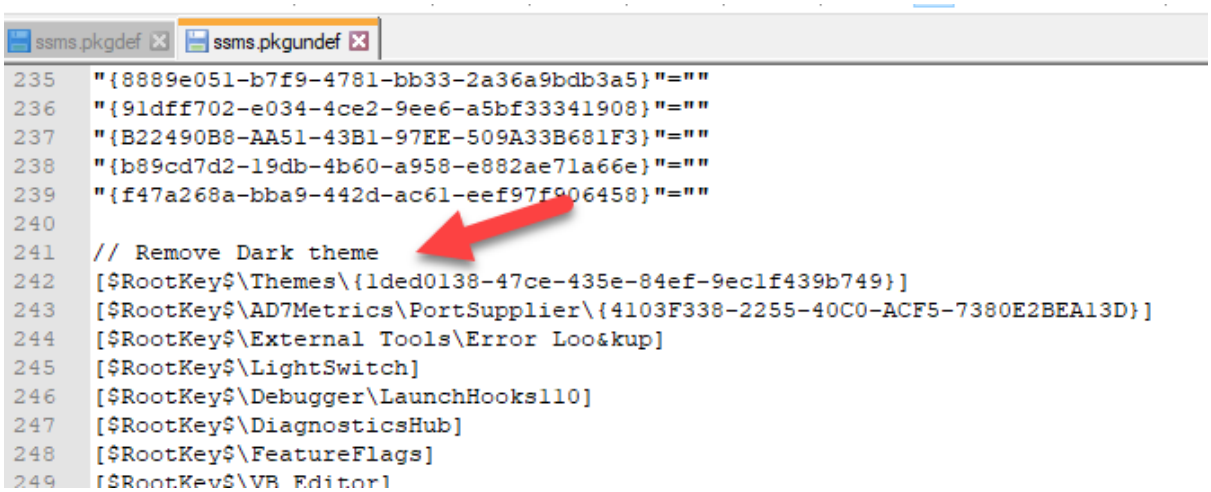
It's basically a list of where things are but note there's also now info about the background telemetry that SSMS performs.

In **ssms.pkgundef,** I can see this:

```
//This file is where you will place registry entries that are to be removed during setup of

// Remove macro support.
//[$RootKey$\Packages\{A659F1B3-AD34-11d1-ABAD-0080C7B89C95}]
//[$RootKey$\Menus]
//"{A659F1B3-AD34-11d1-ABAD-0080C7B89C95}"=""
//[$RootKey$\Projects\{07CD18B1-3BA1-11d2-890A-0060083196C6}]
//[$RootKey$\Projects\{23162FF1-3C3F-11d2-890A-0060083196C6}]
//[$RootKey$\Services\{04BBF6A5-4697-11d2-890E-0060083196C6}]
//[$RootKey$\Services\{55ED27C1-4CE7-11d2-890F-0060083196C6}]
//[$RootKey$\Services\{A659F1B2-AD34-11d1-ABAD-0080C7B89C95}]
//[$RootKey$\ToolWindows\{07CD18B4-3BA1-11D2-890A-0060083196C6}]

// ********
// Remove unused packages
// ********

// TFS SCC Configuration entries.  The TFS entries block Team Explorer from loading.
// Microsoft.VisualStudio.TeamFoundation.VersionControl.HatPackage
[$RootKey$\AutoLoadPackages\{4CA58AB2-18FA-4F8D-95D4-32DDF27D184C}]
// Microsoft.VisualStudio.TeamFoundation.Lab
[$RootKey$\Packages\{17c5d08a-602c-4dfb-82b5-8e0f7f50c9d7}]
// GitHub Package
[$RootKey$\Packages\{c3d3dc68-c977-411f-b3e8-03b0dccf7dfc}]
// Team Foundation Server Provider Package
[$RootKey$\Packages\{5BF14E63-E267-4787-B20B-B814FD043B38}]
```

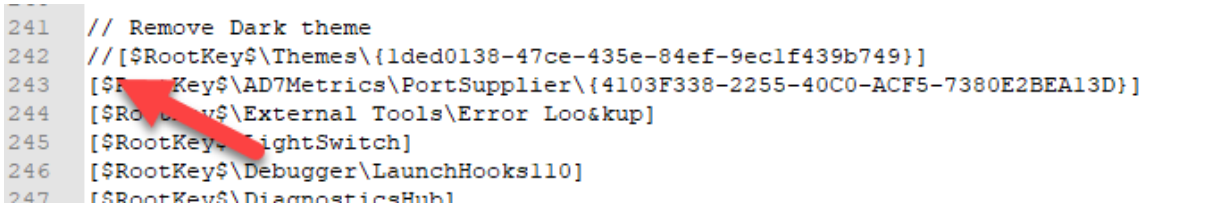
It's basically a list of all the things that are normally there that they have decided to remove. Notice, sadly, that macro support is one of them. I've never understood why we don't have macros. I think they'd be useful.

Regardless, scrolling down we find the culprit for our missing theme:

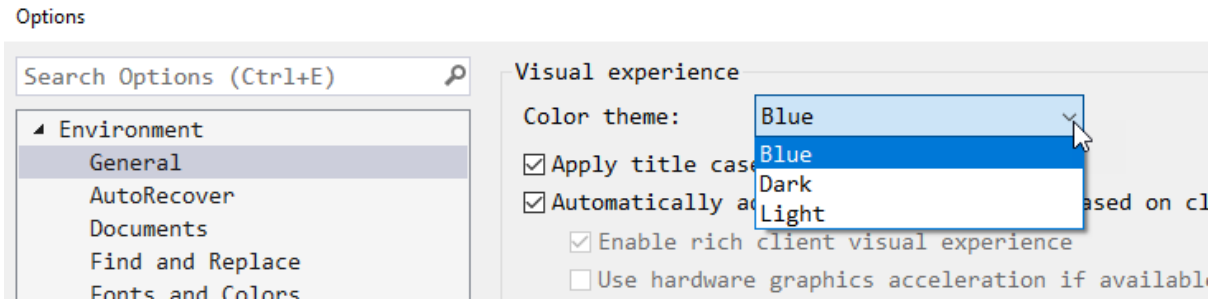



So this is no accident. The team has deliberately decided to remove it. I can only imagine that's because they aren't sure if it will work properly (or know that it won't) and they don't want to have to test or fix it.

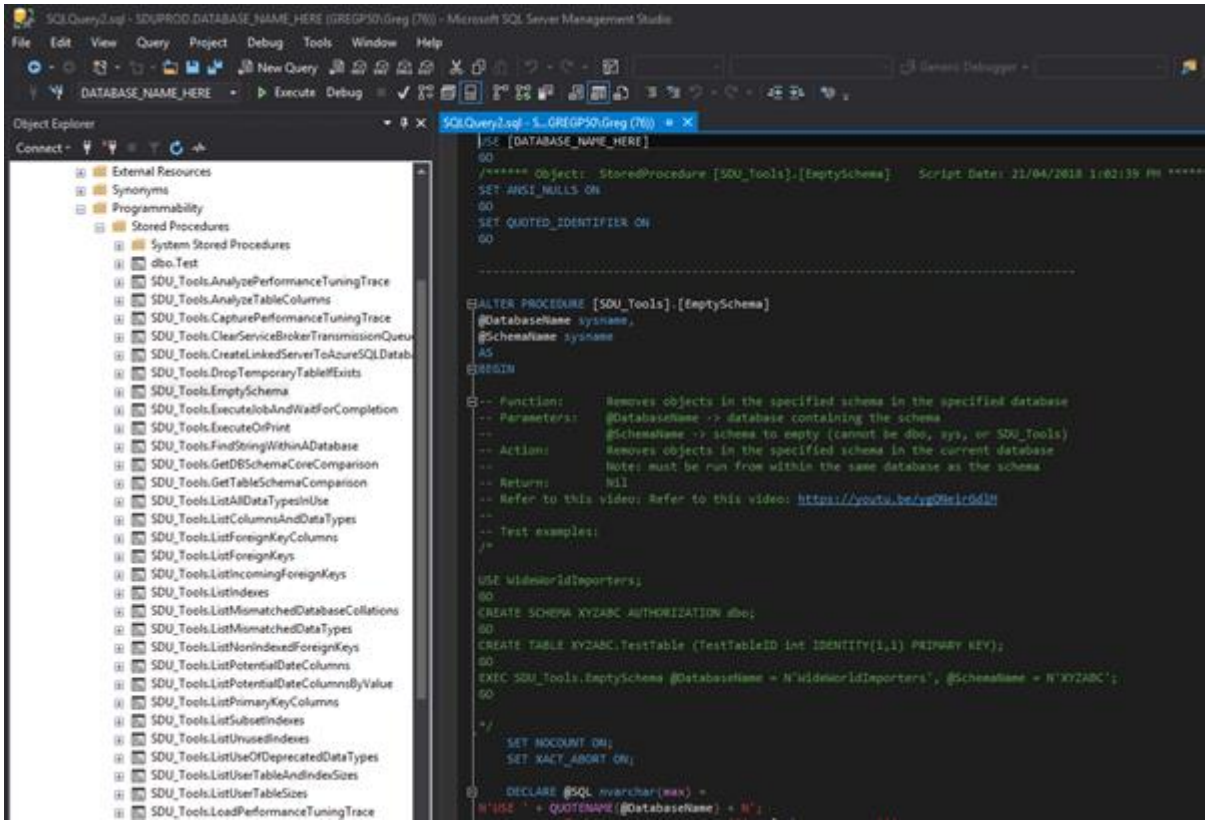
We can enable it, by commenting out that line, and saving it. (We'll have to be an administrator to be able to save it).



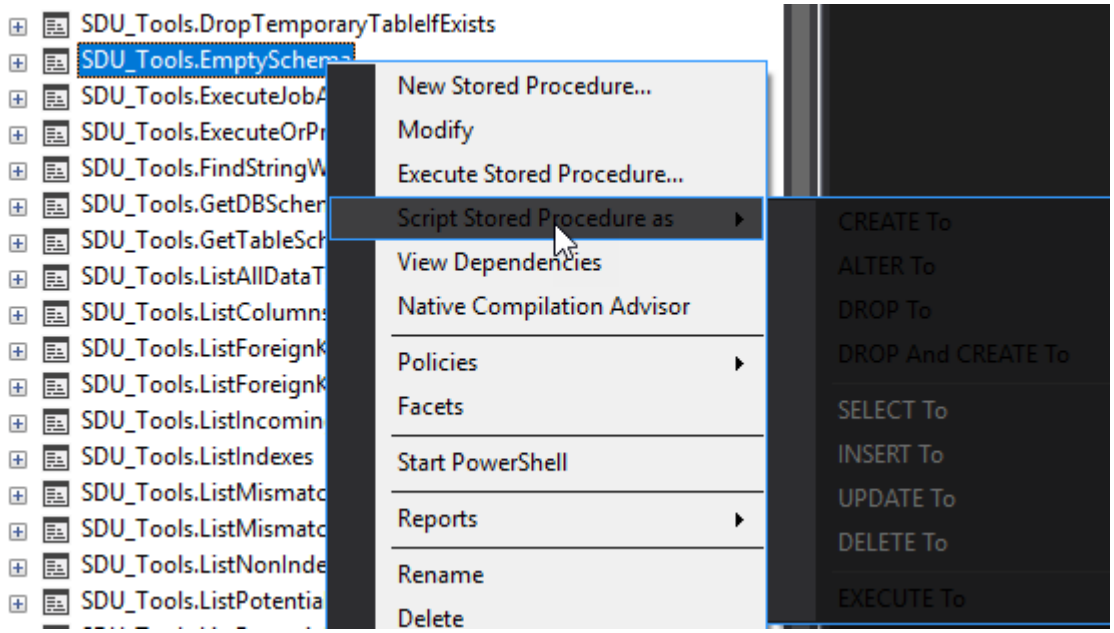

And when we restart SSMS, it's now there:




And notice that it now works:



Well, at least, almost works. It seems fine for editing code, but notice that Object Explorer doesn't play along with a dark theme, nor does many of the items that you launch from within it:




But if you really, really want to edit code with a dark theme all day long, this might be enough for you.

# 3 Query Editing

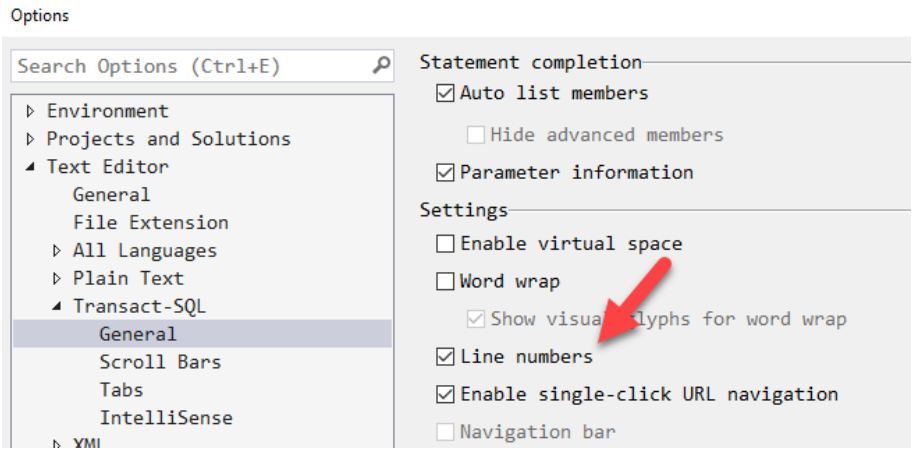
## 3.1 Adding Line Numbers and GOTO Line Number

If you ever have long scripts in SQL Server Management Studio and need to refer to a particular line, it can be helpful to have line numbers shown. This is even more useful if you ever need to write a set of instructions for someone on how to modify a query, or if you are trying to describe how a query works.

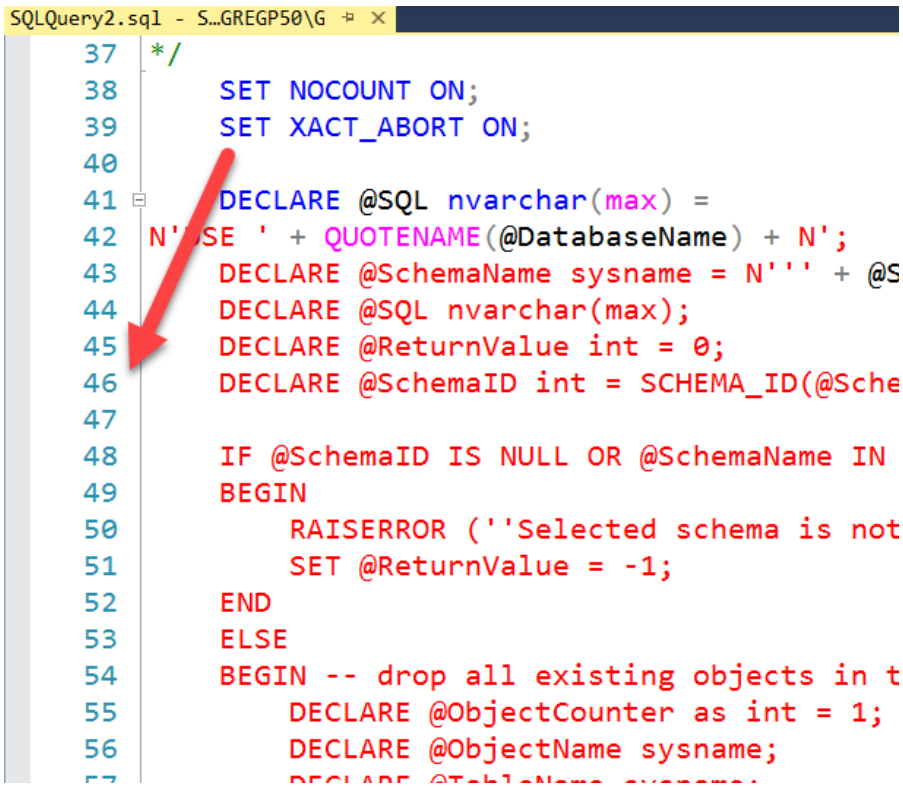
Instead of having a query like this:

```
SQLQuery2.sql - S…GREGP50\G
*/
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @SQL nvarchar(max) =
N'USE ' + QUOTENAME(@DatabaseName) + N';
    DECLARE @SchemaName sysname = N''' + @Schema
    DECLARE @SQL nvarchar(max);
    DECLARE @ReturnValue int = 0;
    DECLARE @SchemaID int = SCHEMA_ID(@SchemaNam

    IF @SchemaID IS NULL OR @SchemaName IN (N''s
    BEGIN
        RAISERROR (''Selected schema is not pres
        SET @ReturnValue = -1;
    END
    ELSE
    BEGIN -- drop all existing objects in the sc
        DECLARE @ObjectCounter as int = 1;
        DECLARE @ObjectName sysname;
        DECLARE @TableName sysname;
        DECLARE @ObjectTypeCode varchar(10);
        DECLARE @ObjectsToRemove TABLE
        (
            ObjectRemovalOrder int IDENTITY(1,1)
```

In Tools, then Options, then Text Editor, then Transact-SQL, on the General tab, there is an option for Line Numbers:



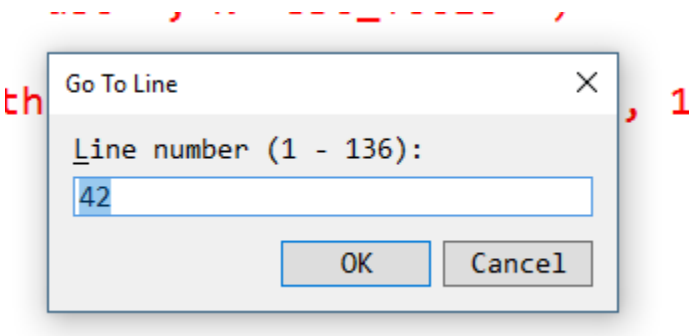
Once that's enabled, the query windows look like this:




The line numbers go in a separate margin. Note that all lines are numbered, unrelated to whether or not they contain an individual SQL statement.

If you have a long script and you need to quickly, reposition the cursor, note that you can also go directly to a line number. On the Edit menu, you can see that Ctrl-G is the shortcut for Go To.

Hitting Ctrl-G pops up a dialog and asks me for a line number:




Then takes me straight there. Note that in the dialog, the maximum line number is shown.

## 3.2 Useful keyboard shortcuts

Visual Studio is a very configurable tool, and particularly in the area of keyboard shortcuts. Because SQL Server Management Studio (SSMS) is based on Visual Studio, it inherits many of these configuration options.

SSMS has a very rich set of keyboard shortcuts. Without trying to cover most of them, I do want to highlight a few that I think are really important to know how to use.

Let's start with an easy set of commands:

You might need to change the case of some values. If I have this code:

```
DECLARE @A_Constant_Value int = 12;
DECLARE @AnotherImportantValue int = 15;
```

I might decide that our new coding standards say that variables that are treated like constants, must be in all capitals and with words separated by underscores. (This is often called SHOUTY_SNAKE_CASE).

I can just highlight the name (or double-click it), the hit Ctrl-Shift-U to make it upper case.

```
DECLARE @A_CONSTANT_VALUE int = 12;
DECLARE @AnotherImportantValue int = 15;
```

And Ctrl-Shift-L would make it lower case.

Now, what if I want to rearrange the order of these declaration lines. If I needed to move the @A_CONSTANT_VALUE line below the other line, I often see people highlight the whole line, then cut it, then paste it below.

A great keyboard shortcut for doing that, is to hit Alt-Shift-T. No matter where you are on the line, it just moves the line down by one.

```
DECLARE @AnotherImportantValue int = 15;
DECLARE @A_CONSTANT_VALUE int = 12;
```

If I wanted to then add a line in between these two lines, what I typically see people do is to put the cursor on the D in the second line (at the beginning of the line), then hit Enter to move it down one, then use the up arrow to go back to the newly emptied line.

What you can do instead is put the cursor anywhere on the first line, and hit Ctrl-Shift-Enter.

```
DECLARE @AnotherImportantValue int = 15;

DECLARE @A_CONSTANT_VALUE int = 12;
```

I'm sure that I'm somewhat anal, and because of this, I also often spend time cleaning up lines. So if I come across a line like the second one here, the spacing grates on me:

```
DECLARE @AnotherImportantValue int = 15;

DECLARE     @A_CONSTANT_VALUE            int =     12;
```

The quickest way to clean up the second line is to highlight the whole line, then hit Ctrl-K followed by Ctrl-Backslash.

```
DECLARE @AnotherImportantValue int = 15;

DECLARE @A_CONSTANT_VALUE int = 12;
```

Notice that it automagically removed all the messy whitespace. It can do multiple lines at once, but it won't rearrange what's on each line.

Finally, if I needed to comment out this code, I'd highlight it and hit Ctrl-K then Ctrl-C.

```
--DECLARE @AnotherImportantValue int = 15;

--DECLARE @A_CONSTANT_VALUE int = 12;
```

And Ctrl-K followed by Ctrl-U will uncomment it.

Two more easy shortcuts: **Ctrl-Home** takes you to the top of the script. **Ctrl-End** takes you to the bottom of the script.

The final most useful keyboard shortcut is **Ctrl-R**. It hides or shows the results pane.

You'll find a detailed list of keyboard shortcuts here:

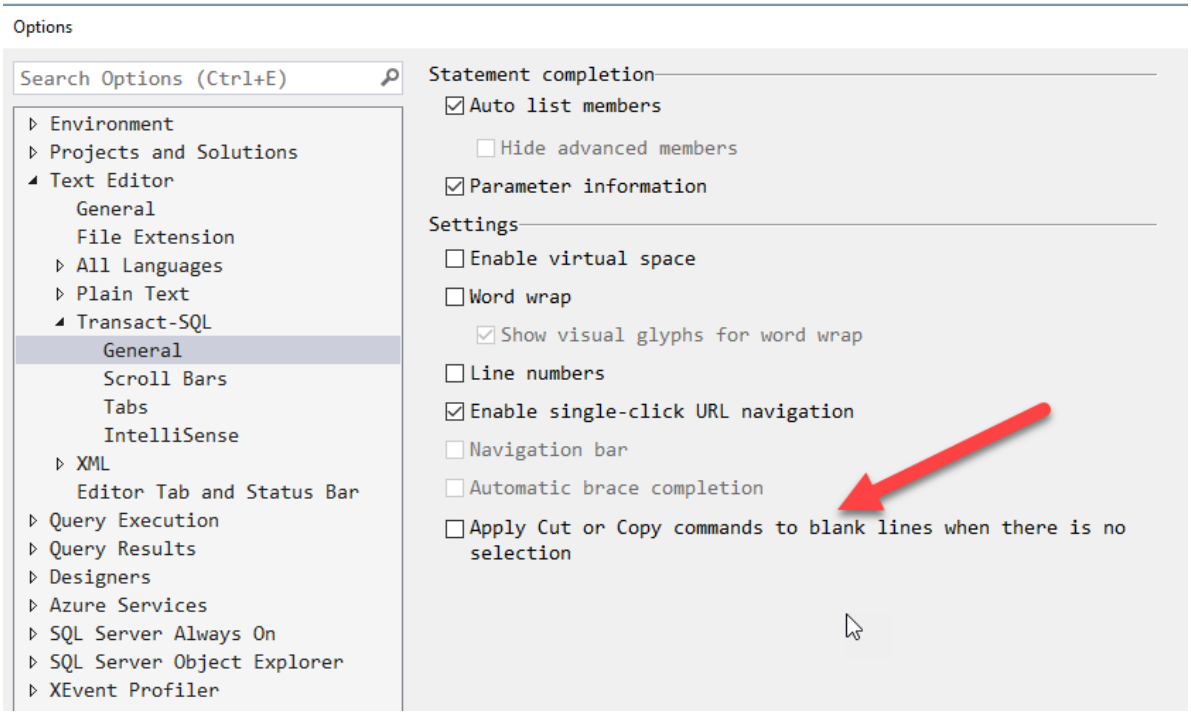
https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-keyboard-shortcuts

### 3.3 Apply cut or copy commands to blank lines when there is no selection

When I'm doing a lot of query editing, I often get a bit mesmerized, particularly if there's a lot of manual copy and paste or cut and paste going on.

One thing that often drives me crazy is when I use Ctrl-C (ie: copy) when I meant to use Ctrl-V (ie: paste). Invariably, I do this when I have nothing highlighted at all. So not only did I not get the value pasted, I just copied an empty value into the clipboard.

But SQL Server Management Studio (SSMS) has your back on this. (And so does Visual Studio)




In Tools, then Options, then Text Editor, then Transact-SQL, under the General tab, there's an option for **Apply Cut of Copy commands to blank lines when there is no selection**.

The default is that it works as expected (like you don't want it to), but if you uncheck it, you might have just saved yourself some annoyance.

If I'm working with this code:

```
DECLARE @AnotherImportantValue int = 15;

DECLARE @A_CONSTANT_VALUE int = 12;
```

and I highlight the word DECLARE and hit Ctrl-C, then move to the blank line and hit Ctrl-C instead of Ctrl-V, I'd have just lost my first clipboard item. With this option unchecked, I can just smile, and hit Ctrl-V again, and it will still paste:

```
DECLARE @AnotherImportantValue int = 15;
DECLARE
DECLARE @A_CONSTANT_VALUE int = 12;
```

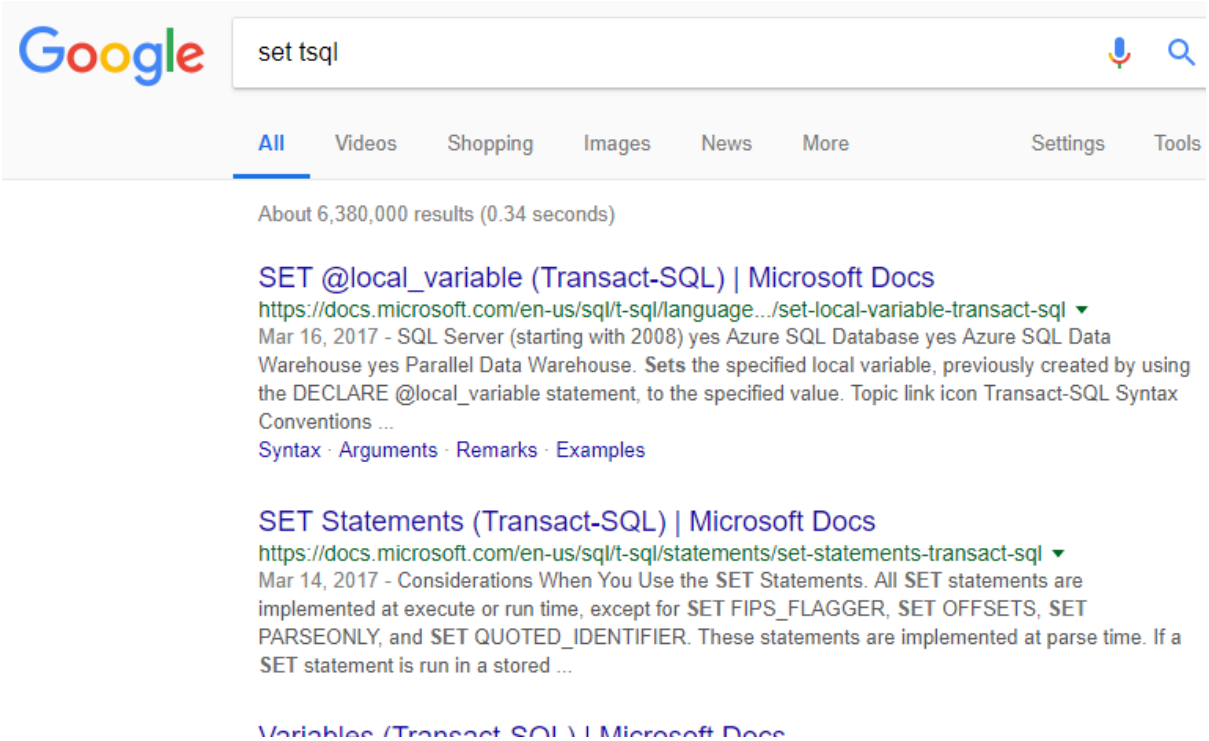
### 3.4 The magical F1 key – help on syntax and metadata

I used to always recommend that people install Books Online (BOL) on their systems. It's ironic that it was called "Online", given we're really talking about "Offline" nowadays, but back when we first were talking about it, we were comparing it to a physical book, not to a live reference on the Internet.

Nowadays though, I find that the version online is so far superior to the one that you can install locally, that I think it's better to just use the online version. I particularly like the way that the online books are now cross-version ie: each page covers all supported versions, instead of having a separate page for each version.

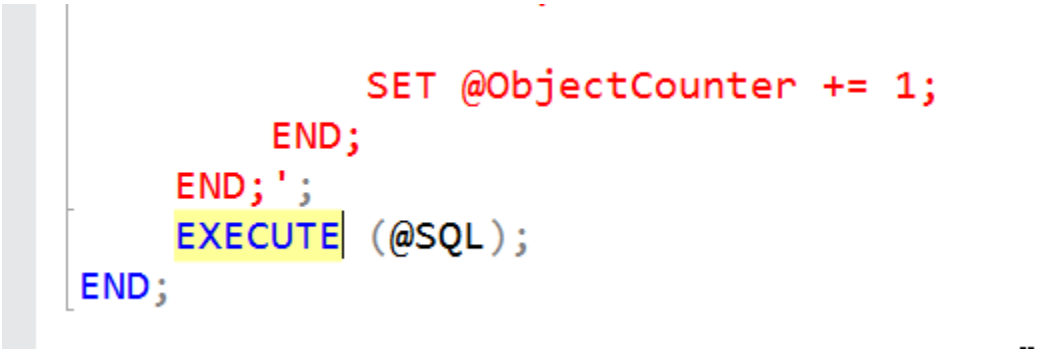
Given there is a lot of information online, what's the quickest way then to find the page that you're after?

For T-SQL commands, I find the best option is to type the command name followed by tsql into the search bar. For example, here's a search for the SET command:

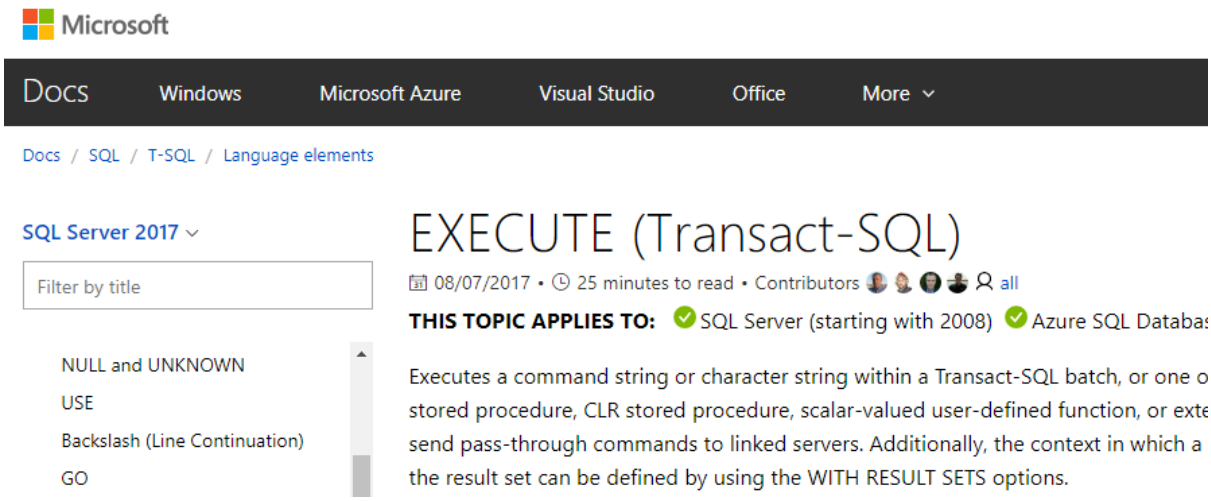



Invariably, the command that you're after will be either the first or second entry.

But there's an even quicker way when you're editing in SQL Server Management Studio (SSMS). In this script that I'm editing:



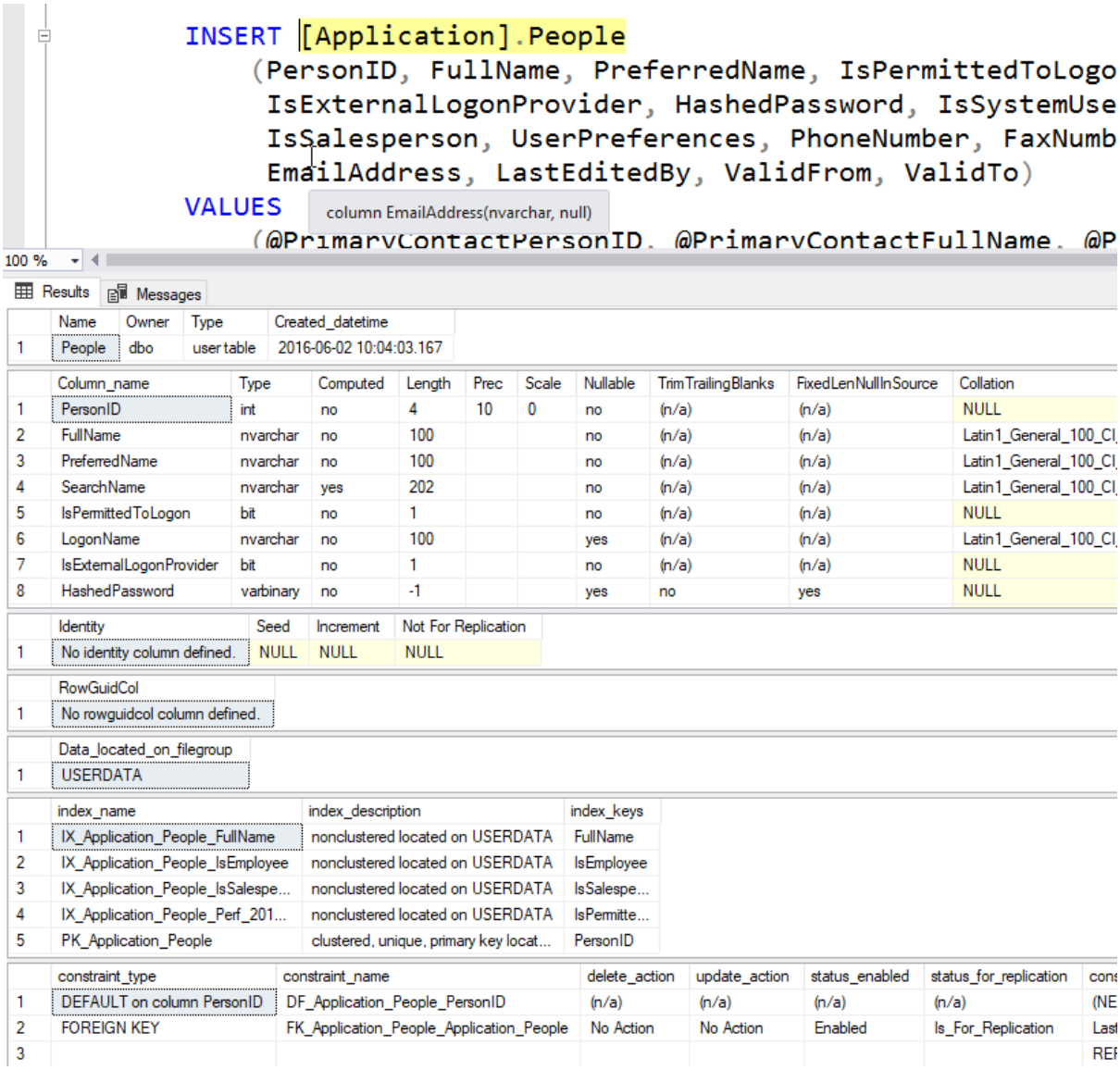
If I double-click EXECUTE to highlight it, then hit F1, I'm taken directly to the correct page.




That's awesome but the F1 key can do more. If I was looking at or editing this script from WideWorldImporters:

```
BEGIN TRAN;

INSERT [Application].People
    (PersonID, FullName, PreferredName, IsPermittedToLogon, Logon
     IsExternalLogonProvider, HashedPassword, IsSystemUser, IsEmp
     IsSalesperson, UserPreferences, PhoneNumber, FaxNumber,
     EmailAddress, LastEditedBy, ValidFrom, ValidTo)
VALUES
    (@PrimaryContactPersonID, @PrimaryContactFullName, @PrimaryCo
     0, NULL, 0, 0,
     0, NULL, N'(' + CAST(@AreaCode AS nvarchar(20)) + N') 555-01
```

and I'm wondering about the People table, I can highlight the full table name [Application].People, then hit **Alt-F1**.I'm then returned all sorts of information about the table:




What SSMS is doing is running the **sp_help** command for that object. In another post, we'll talk about how you could change that if needed, or change what happens with other function keys.

### 3.5 Fixing or improving Books Online

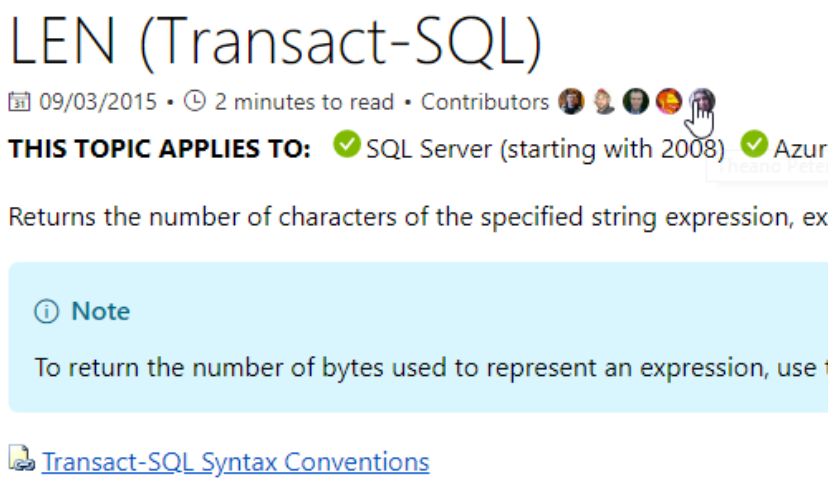
I mentioned in an earlier post that I think the online version of Books Online (BOL) is now superior to the version that you can install locally.

I particularly like the way that the online books are now cross-version ie: each page covers all supported versions, instead of having a separate page for each version.

But one of the really big bonuses is that you now have the opportunity to change the documentation if you think it's incorrect or you think it could be improved. Microsoft have placed all the documentation in a Git repository and you can change it. Doing so is easier than you might expect.

Let's look at an example.

I've searched for the LEN function in T-SQL and found the page:


LEN (Transact-SQL)

09/03/2015 • 2 minutes to read • Contributors

**THIS TOPIC APPLIES TO:** SQL Server (starting with 2008) Azur

Returns the number of characters of the specified string expression, ex

Note

To return the number of bytes used to represent an expression, use

Transact-SQL Syntax Conventions

## Syntax

```
LEN ( string_expression )
```

Note the small list of faces appearing under the command name. These are people who have contributed to this page.

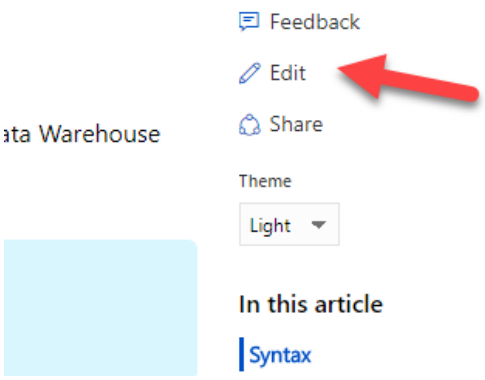
Now while I'm reading the page, I see this:


## Examples

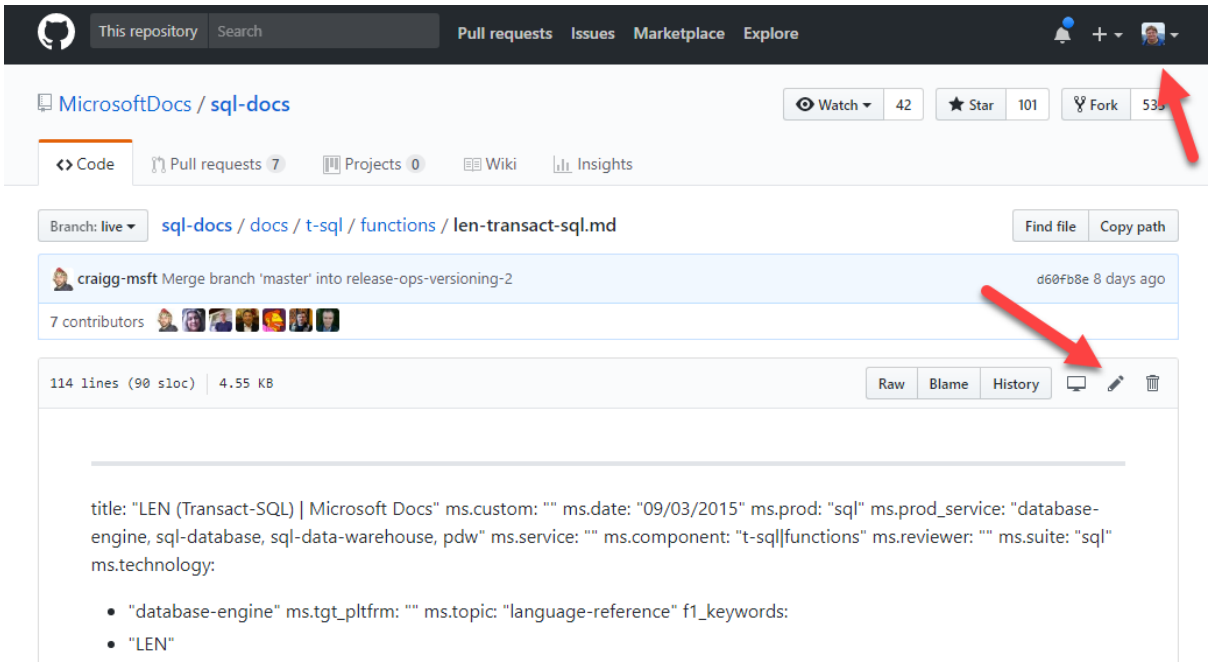
The following example selects the number of characters and the data in `FirstNa` AdventureWorks2012 database.

```
SELECT LEN(FirstName) AS Length, FirstName, LastName
FROM Sales.vIndividualCustomer
WHERE CountryRegionName = 'Australia';
GO
```

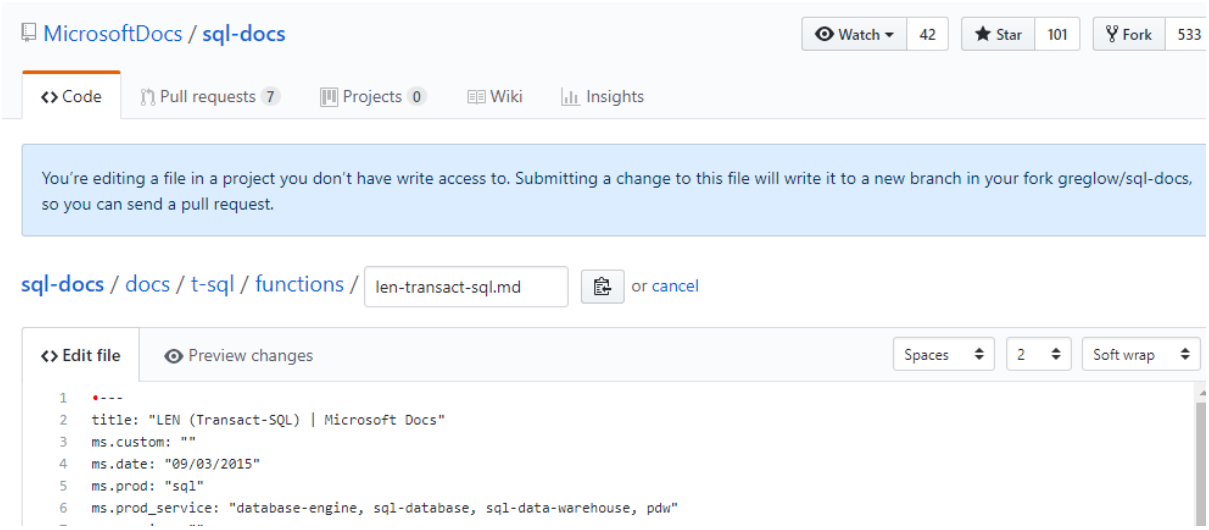
I'm puzzled why that example is specific to AdventureWorks2012. That code would work on all AdventureWorks versions. So let's try to change it. At the top of the page, there's an **Edit** link.




I'll click this and I'm taken to the page in Git:



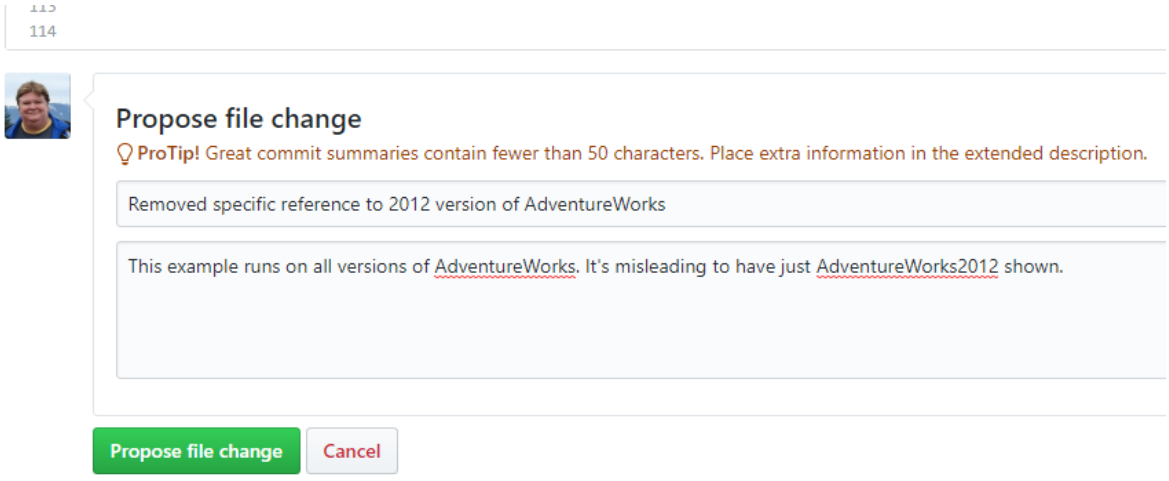

Note in the top right-hand corner that I've already logged onto Git. Also notice the Edit pencil. Now I can't just directly change this info, so what I do is click this to "fork" the project so that I have my own copy to edit.



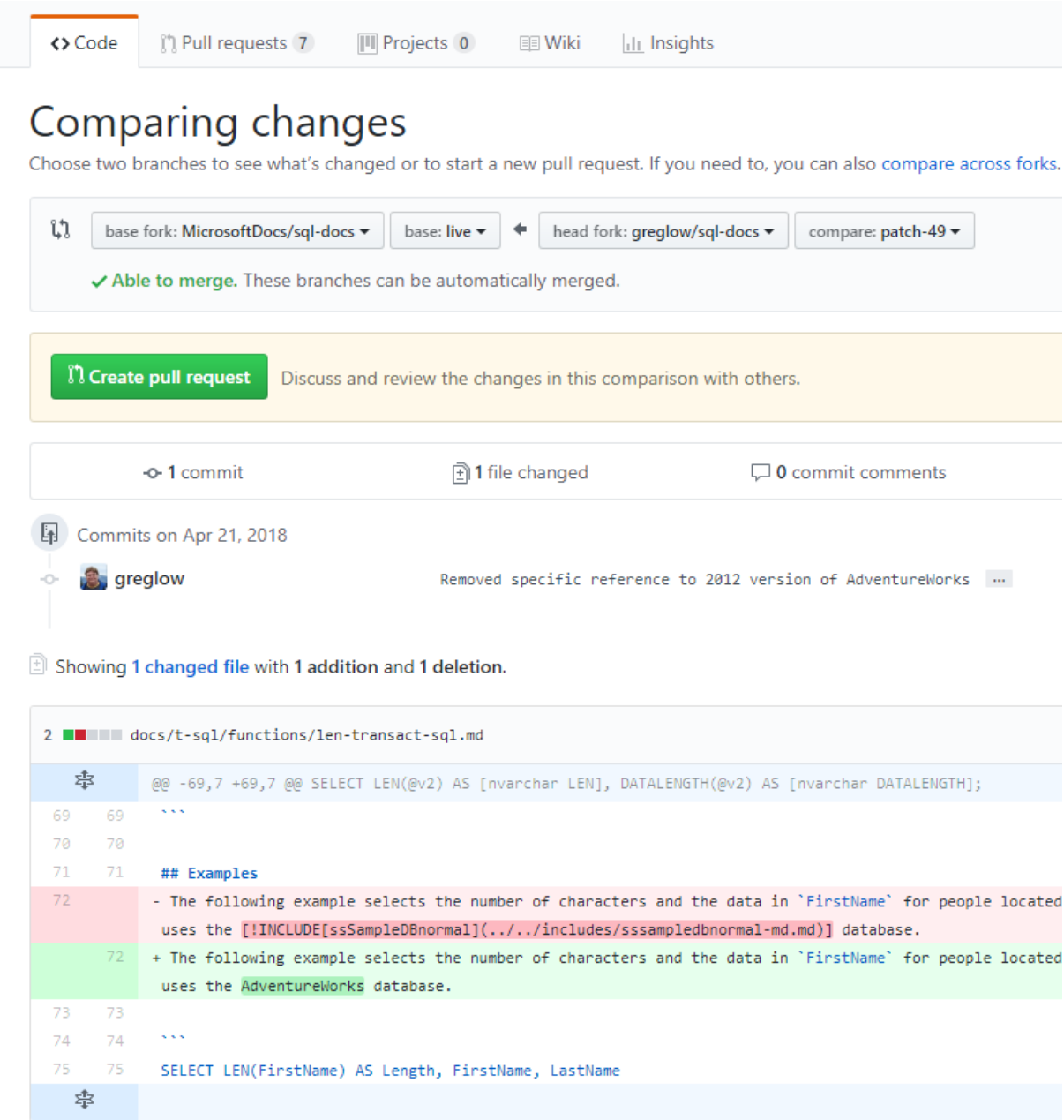

Now I can make the change that I want to:

```
70
71  ## Examples
72   The following example selects the number of characters and the data in `FirstName` for people loca
    the AdventureWorks database.
73
74  ```
75  SELECT LEN(FirstName) AS Length, FirstName, LastName
76  FROM Sales.vIndividualCustomer
77  WHERE CountryRegionName = 'Australia';
78  GO
79  ```
80
```

And at the bottom of the page, I explain why:

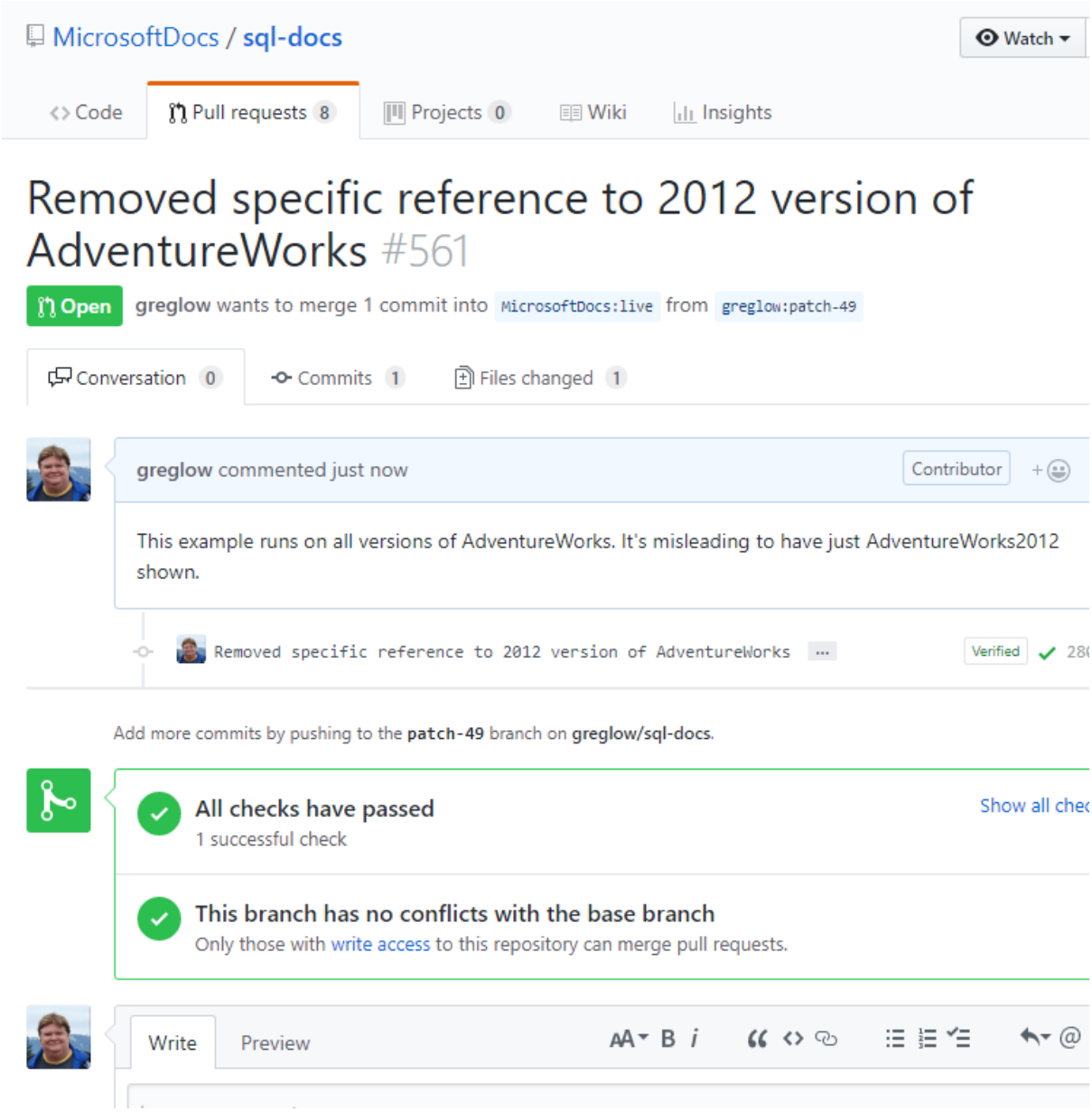


Then I click **Propose file change**, and I'm taken to a page that asks me to **create a pull request**.




Git runs some automated checks, and makes sure that I'm not suggesting a change that can't easily be merged.

If I'm happy with the differences shown, I just click **Create pull request** and again another time. This page is then sent off to the person who is responsible for maintaining the page:




If they agree, your change will be merged in. Either way, you'll receive emails telling you what's going on.

If you have added a lot of changes or code, you'll also receive another email asking you to agree to be a contributor.

This is a great new option and once you're used to it, very easy to do.

## 3.6 Manually prompting for and Refreshing Intellisense

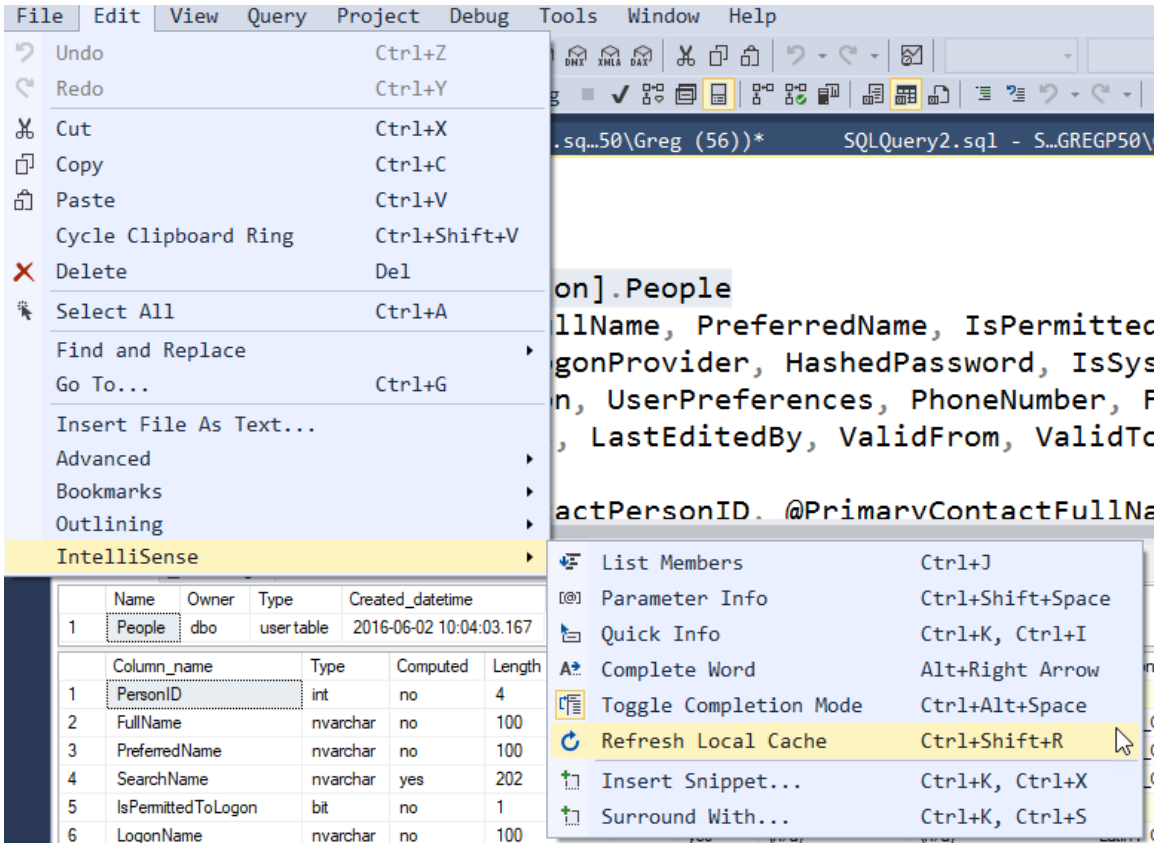
Intellisense is one of the best things that's ever been added to Visual Studio or to SQL Server Management Studio (SSMS). It's hard to remember back to before it was added, or how we worked then.

I had a young friend from the United Kingdom who had just completed a Computer Science degree and one of the things that he was most proud of, is that he knew so many HTML tags and which attributes went with which tags. When I showed him HTML Intellisense in Visual Studio, I think he was about to cry.

While Intellisense in SSMS pretty much works as expected, there are a few things that can go wrong to confuse it.

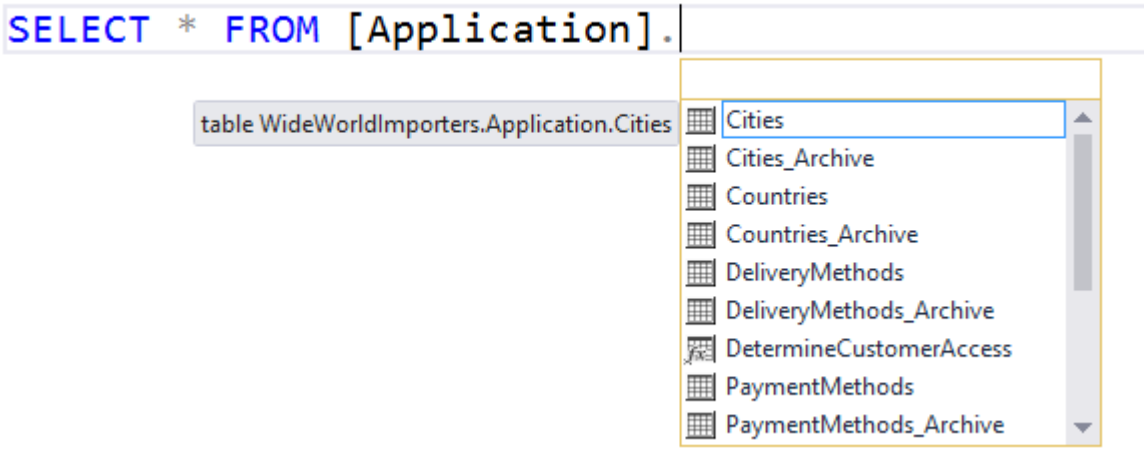
The first problem is that it caches the list of objects to make sure it can perform quite well. But the cache can get out of sync with reality. A common cause of that is if I execute T-SQL commands in one query window, and I'm using those same objects in another window.

I've seen people quite puzzled about this but it's easy to fix. On the Edit menu, you can see that the shortcut key to refresh the local cache is Ctrl-Shift-R:




So if you see a bunch of unexpected "red squigglies", first thing to do is to hit **Ctrl-Shift-R**.

Another thing that can happen is that you get into a situation where the prompted values won't appear. If I type the following code:




note that the Intellisense has appeared. But if I hit the Escape key, it will disappear again. So users wonder how to get it back. Now one option is to backspace over the period, then type the period again. The standard option though, is to hit **Alt-RightArrow**.

An alternative to this is to hit **Ctrl-Space**, and that's easier to hit anyway.

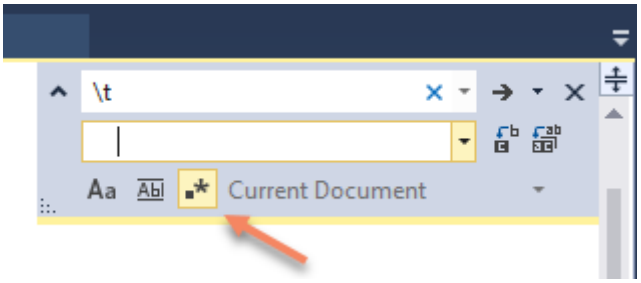
### 3.7 Replace Tabs with Spaces and do Macro-like work in SSMS using Regular Expressions

A request that I hear all the time, is:

**"I don't like tabs but <insert name of annoying colleague here> decided he likes to use them. How do I remove them?"**

Now SSMS allows you to choose to use spaces instead of tabs (in Tools > Options) but that doesn't fix existing tabs.

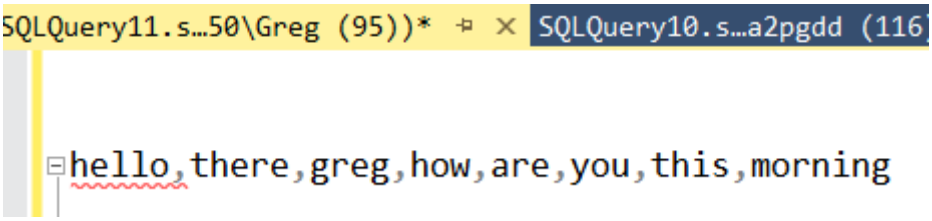
The regular expression functions in SSMS let you do this. Hit Ctrl-H to open the Find and Replace dialog (or use the Edit > Find and Replace > Quick Replace menus), then configure the window as follows:



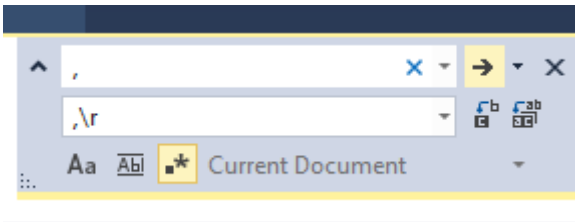

Note that I've clicked the little square and asterisk thing (which says to use regular expressions), then put the change from text to \t which is a tab and set the change to text to four spaces (or however many you want), then I click the Replace All button and magically all my tabs are replaced.

But this is just the start of what we can do.

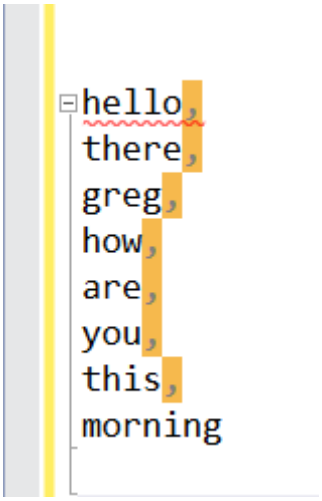
For example, if I have a long row of comma-delimited values like this:




Now I could go to each comma and hit enter after each one. That's what I see most people doing. But you could do this:



I'm changing a comma to a comma followed by a carriage return. And magically, this is the outcome:

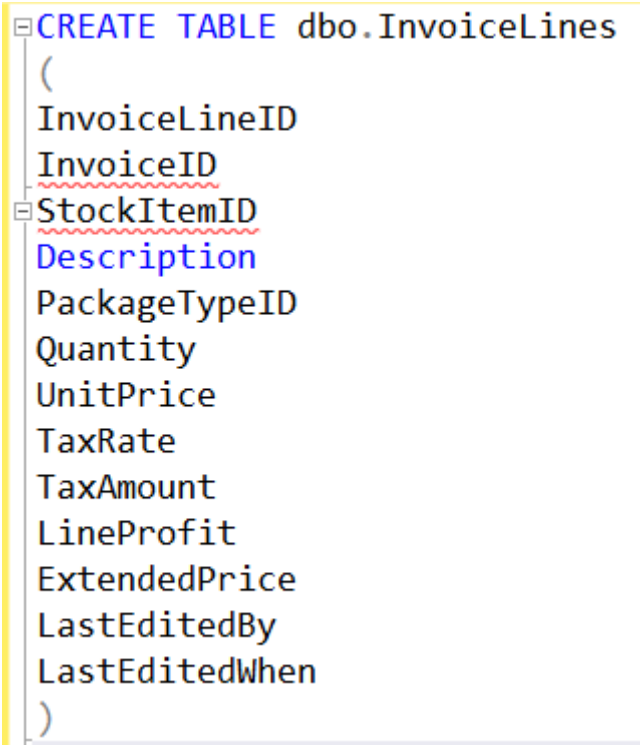



There are so many things you can use this regular expression functionality for within SSMS. I encourage you to try it out.

### 3.8 Selecting and modifying rectangular regions in SSMS

I often see people using SQL Server Management Studio (SSMS) and doing very repetitive editing tasks that could easily be carried out by using the selection and changing of rectangular regions of code.

The simplest example of doing this is to insert a bit of text on a number of rows. Take the following code as an example:




I've got the skeleton of a list of columns in a CREATE TABLE statement but let's assume that I'm a **"comma in front"** person and want to put a few spaces and a comma, etc. in front of each column after the second.

I could just put the cursor on the InvoiceID line and type what I wanted, then do the same again on the next line. I could do it on the first line, then select and copy it, and insert it into the front of every other line.

What I should do, however, is put the cursor in front of InvoiceID, and while holding Alt-shift, use the down arrow to select the beginning of every line, then just type what I want.

Similarly, if I want to change text on a bunch of lines, I can do the same by using both down arrow and side arrow to select a rectangular region, then just type to replace. In this example, I have a table name that I want to change from InvoiceLines to OrderLines:

```
Somedatabase.InvoiceLines.InvoiceID
Somedatabase.InvoiceLines.StockItemID
Somedatabase.InvoiceLines.Description
Somedatabase.InvoiceLines.PackageTypeID
Somedatabase.InvoiceLines.Quantity
Somedatabase.InvoiceLines.UnitPrice
Somedatabase.InvoiceLines.TaxRate
Somedatabase.InvoiceLines.TaxAmount
Somedatabase.InvoiceLines.LineProfit
Somedatabase.InvoiceLines.ExtendedPrice
Somedatabase.InvoiceLines.LastEditedBy
Somedatabase.InvoiceLines.LastEditedWhen
```

So I select the starting point, Alt-shift and arrows to select the rectangle, then just type OrderLines.

### 3.9 Using the Clipboard Ring in SSMS

Two key combinations used by SQL Server T-SQL developers every day are Ctrl-C and Ctrl-V for copy and paste.

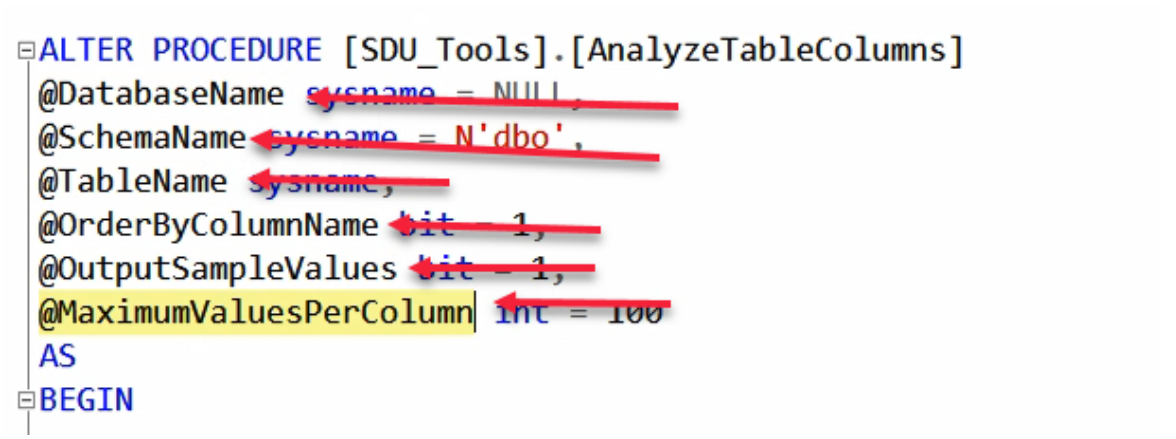
But many users of SQL Server Management Studio (SSMS) don't realize that it has a clipboard ring and can deal with several objects in the clipboard at the same time.

Let's see an example.

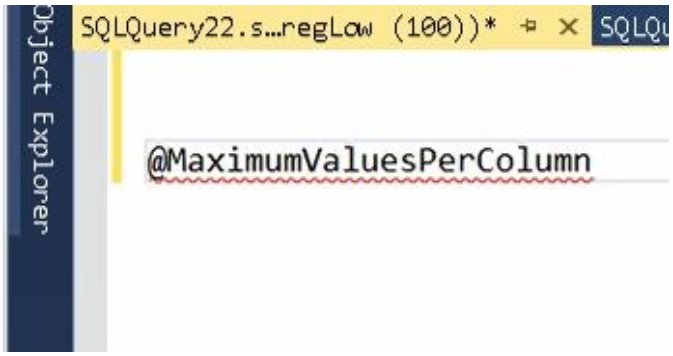
In this screen shot, I've opened a query window with the source code of the AnalyzeTableColumns procedure from SDU_Tools.

```
ALTER PROCEDURE [SDU_Tools].[AnalyzeTableColumns]
@DatabaseName sysname = NULL,
@SchemaName sysname = N'dbo',
@TableName sysname,
@OrderByColumnName bit = 1,
@OutputSampleValues bit = 1,
@MaximumValuesPerColumn int = 100
AS
BEGIN

-- Function:      Analyze a table's columns
-- Parameters:    @DatabaseName sysname            -> (default current database)
--                @SchemaName sysname              -> (default dbo) schema for t
```

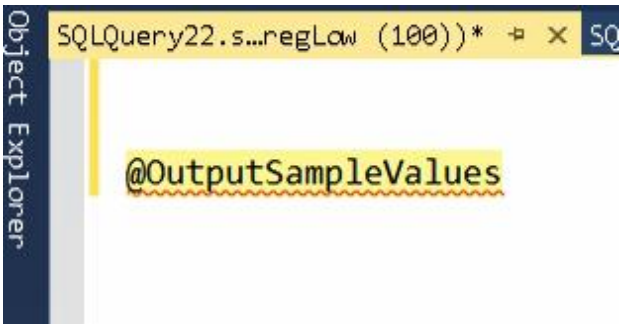
I might want to work with the parameters for that procedure, so I double-click and hit Ctrl-C for each of the parameter names.



I've then opened a new query window where I want to work. If I hit Ctrl-V, I just get the last value that I copied, as expected:




However, instead of using Ctrl-V, if I use Ctrl-Shift-V, I see the same value, but if I continue to hit Ctrl-Shift-V, I see the previous clipboard entries, one at a time. I can then use one or more of the other parameter values that I copied before:




This is one of those things that once you get used to it, you'll wonder how you ever worked without it.

### 3.10 Code Outlining

For some years now, SQL Server Management Studio (SSMS) has had the ability to use code outlining, the same way that other Visual Studio applications can.

This can be very useful when you are trying to navigate around a large script file.

The simplest usage is to collapse or expand a region of code. Note that in the following script, code regions have been automatically added by SSMS:

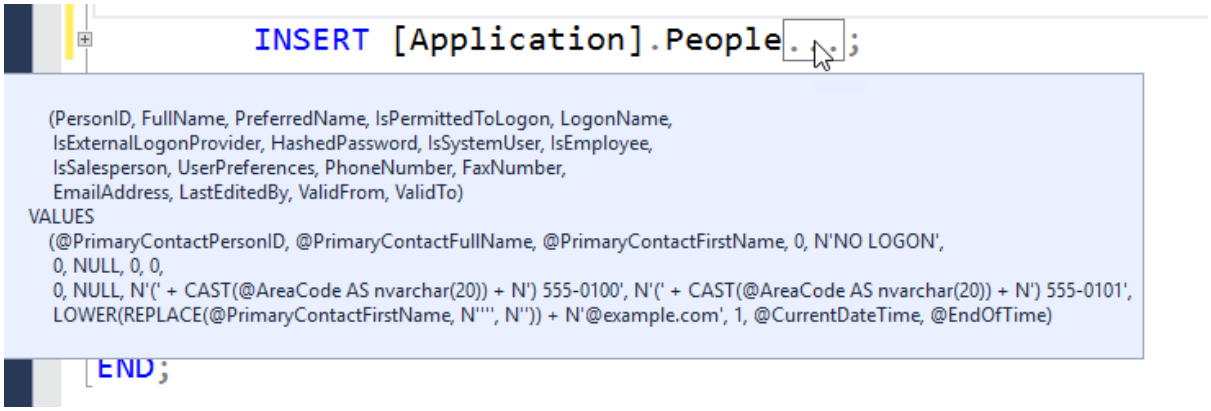
```
        SET @PostalPostalCode = @DeliveryPostalCode;

        SET @PrimaryContactPersonID = NEXT VALUE FOR Sequences.PersonID

        BEGIN TRAN;

        INSERT [Application].People
            (PersonID, FullName, PreferredName, IsPermittedToLogon, Log
             IsExternalLogonProvider, HashedPassword, IsSystemUser, IsE
             IsSalesperson, UserPreferences, PhoneNumber, FaxNumber,
             EmailAddress, LastEditedBy, ValidFrom, ValidTo)
        VALUES
            (@PrimaryContactPersonID, @PrimaryContactFullName, @Primary
             0, NULL, 0, 0,
             0, NULL, N'(' + CAST(@AreaCode AS nvarchar(20)) + N') 555-
             LOWER(REPLACE(@PrimaryContactFirstName, N'''', N'')) + N'@

        INSERT Sales.Customers
            (CustomerID, CustomerName, BillToCustomerID, CustomerCatego
```

This allows us to click on the outline handles, and collapse the code:

```
        SET @PostalPostalCode = @DeliveryPostalCode;

        SET @PrimaryContactPersonID = NEXT VALUE FOR

        BEGIN TRAN;

        INSERT [Application].People...;

        INSERT Sales.Customers...;

        COMMIT;

        SET @Counter += 1;
    END;
END;
```

Note that when the region of code is collapsed, the name of the region is shown as the first line of the code within the region, truncated.

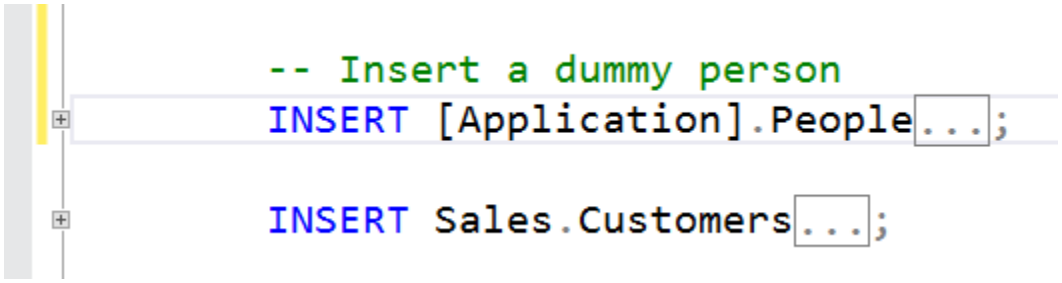
If you hover over the ellipsis (the dot dot dot) at the beginning of the code region, you'll be shown what's contained within the region:



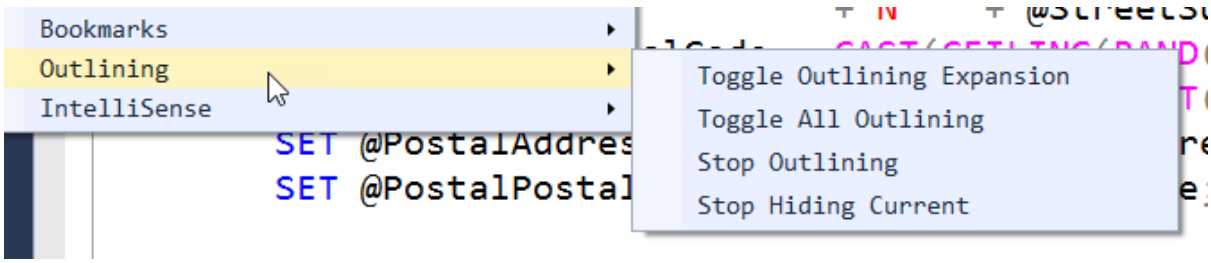

Now, what's missing?

I'd love to be able to just drag regions around.

I'd also love to be able to name the regions better. It's not too bad if the regions are procedures or functions but for other chunks of code, there's really no good option. Note that if I add a comment immediately above the code, it's not part of the same region. It might be better if it was like that, or if a specific comment could be treated as a region heading:




In the Edit menu, the Outlining submenu doesn't show anything else useful at this point, apart from bulk operations:



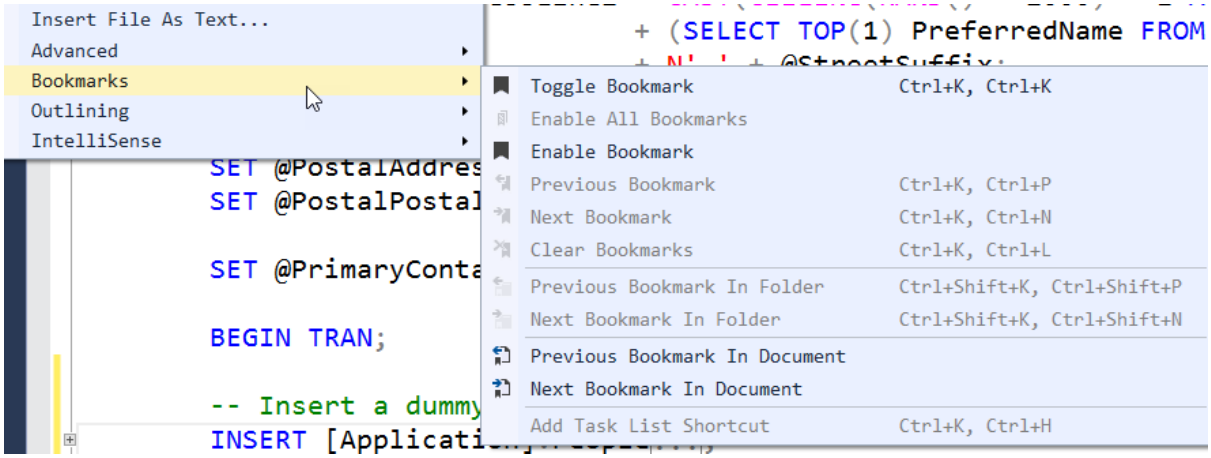
### 3.11 Using Bookmarks

In another section, I was discussing how outlining can be helpful with navigating around within a large T-SQL script file.

If you were trying to do that within a Microsoft Word document, the most common thing to use is **bookmarks**, and SQL Server Management Studio (SSMS) has them as well.

Bookmarks are simply placeholders within a script. (They can also apply to other types of document within SSMS). Where I find them very useful is when I'm working in two or three places within a long script at the same time. Perhaps I'm working on a function, and on the code that calls the function. By using bookmarks, I'm not flipping endlessly around the script file, and can jump directly from placeholder to placeholder.

A quick check of the Bookmarks submenu (under the Edit menu), shows what's available:




You toggle (enable or disable) a bookmark at a point, by using **Ctrl-K and Ctrl-K**. You can then navigate forwards or backwards using **Ctrl-K and Ctrl-N** (next), or **Ctrl-K and Ctrl-P** (previous).

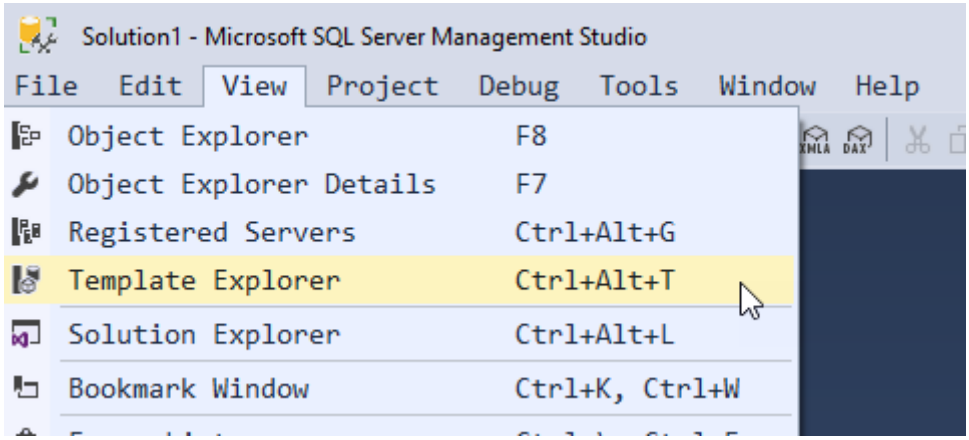
Note that there are options available for both the document and the folder. The folder option can be particularly powerful.

Bookmarks are an often-ignored but highly useful part of using SSMS. If you don't currently use them, you might want to consider them.

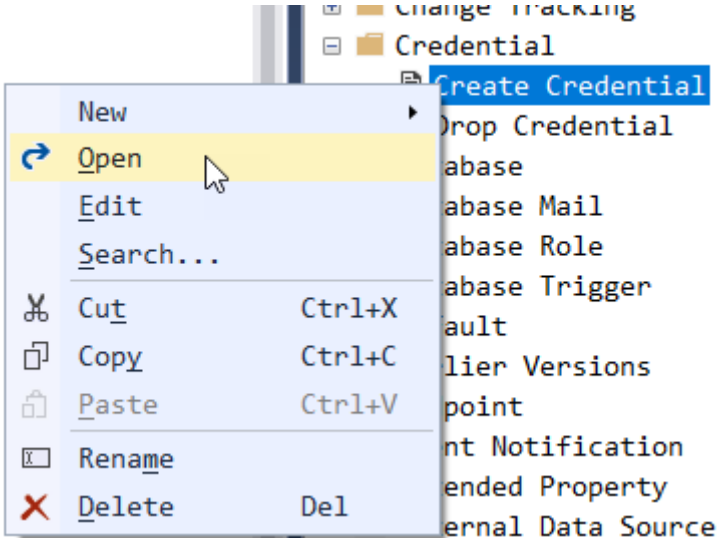
## 3.12 Using Templates

Templates are easy to use because they don't appear on the screen by default, many people don't even realize that they are there. Let's look:

From the View menu, you can choose to view Template Explorer:

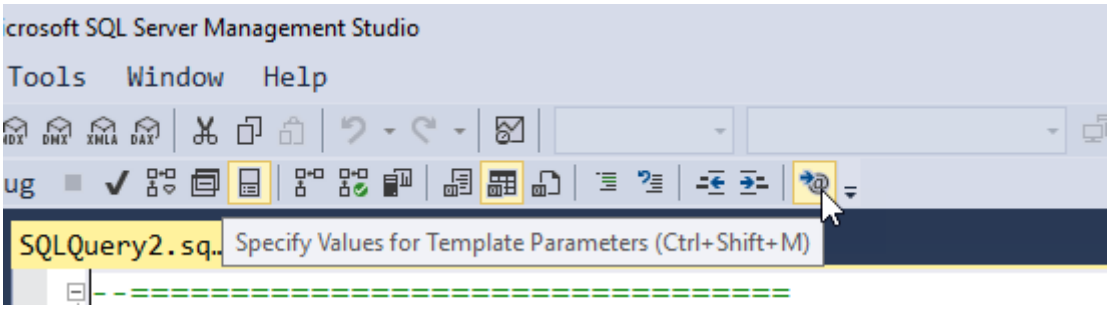



When Template Explorer opens, you see a list of predefined templates, plus any that you have created yourself. (One advantage of templates is that they are quite easy to create). In this case I've expanded the **Credential** folder, right-clicked the **Create Credential** template, then clicked **Open**.

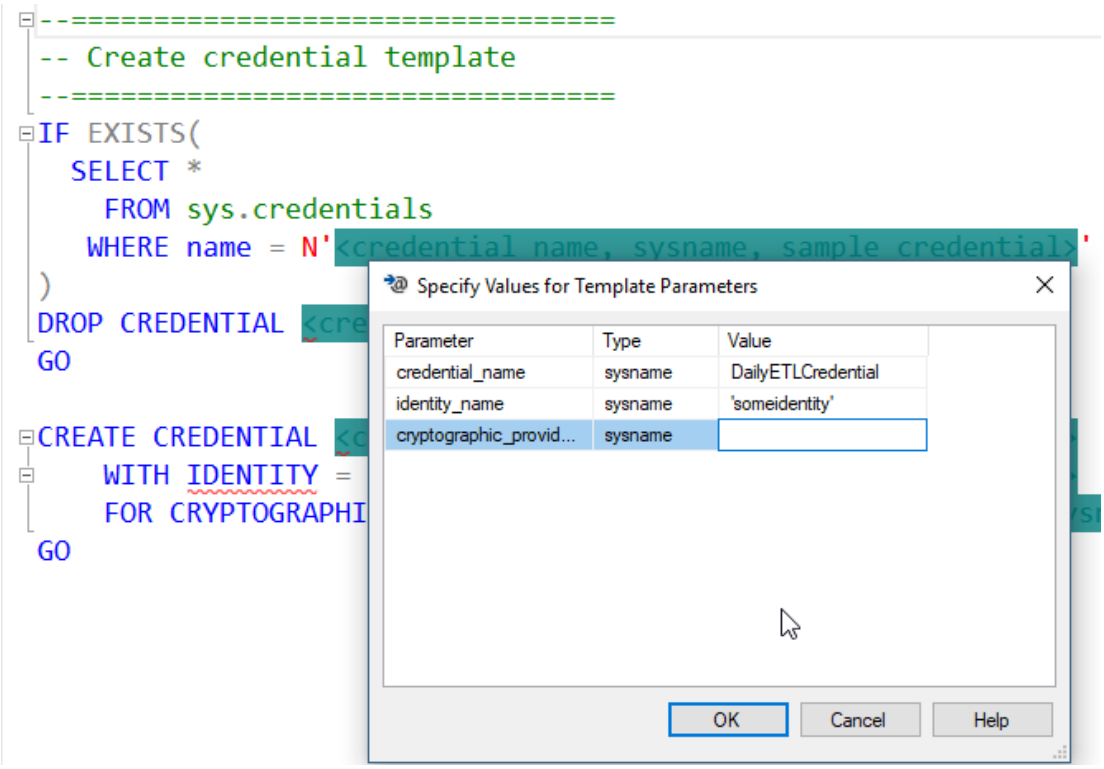



This then opens a new query window with the template for creating a credential inserted.

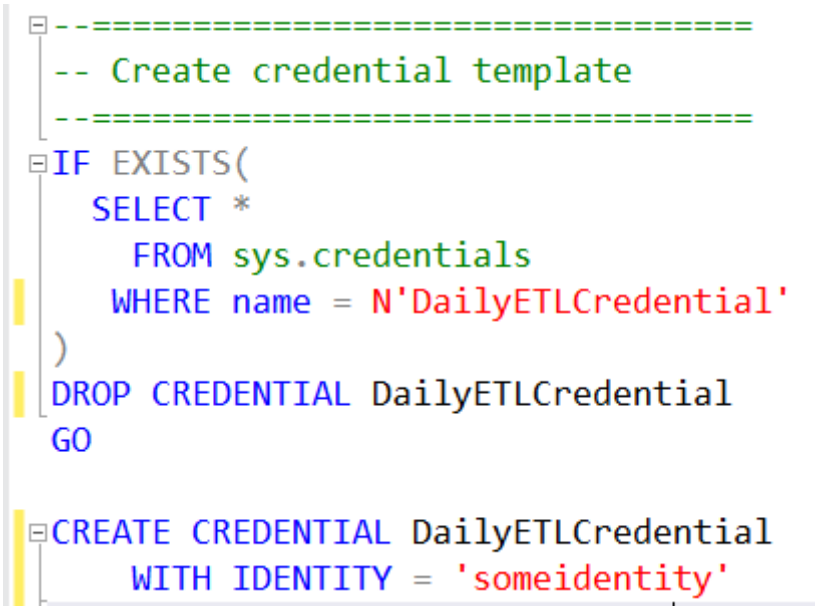
At this point, it's tempting to start editing the template manually, but instead click the **Specify Values for Template Parameters** button on the toolbar.




This finds all the declared parameter values, and opens a window where you can enter the values.




By completing the values in this window, you avoid the need to have to put the same value in many places. In this example, the name of the credential is used in several places.



```
--==================================
-- Create credential template
--==================================
IF EXISTS(
  SELECT *
    FROM sys.credentials
   WHERE name = N'DailyETLCredential'
)
DROP CREDENTIAL DailyETLCredential
GO

CREATE CREDENTIAL DailyETLCredential
    WITH IDENTITY = 'someidentity'
```

Templates are a great way to work with predefined code layouts, more than you can do with snippets. SQL Server Management Studio comes with a large number of T-SQL templates but this feature becomes even more powerful when you create your own.

Later, I'll show you how to create your own snippets and templates.

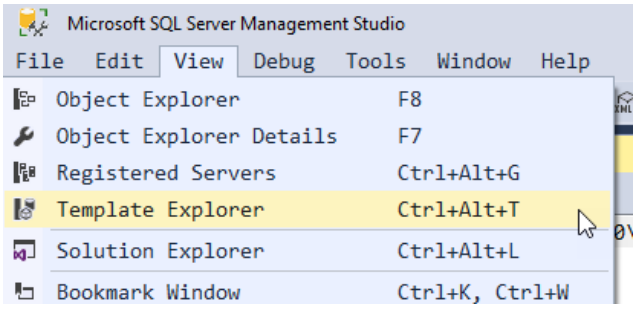
## 3.13 Creating T-SQL Templates in SQL Server Management Studio (SSMS)

In another section, I mentioned how useful templates can be. I said that I'd discuss how to create them later. Well that's today.

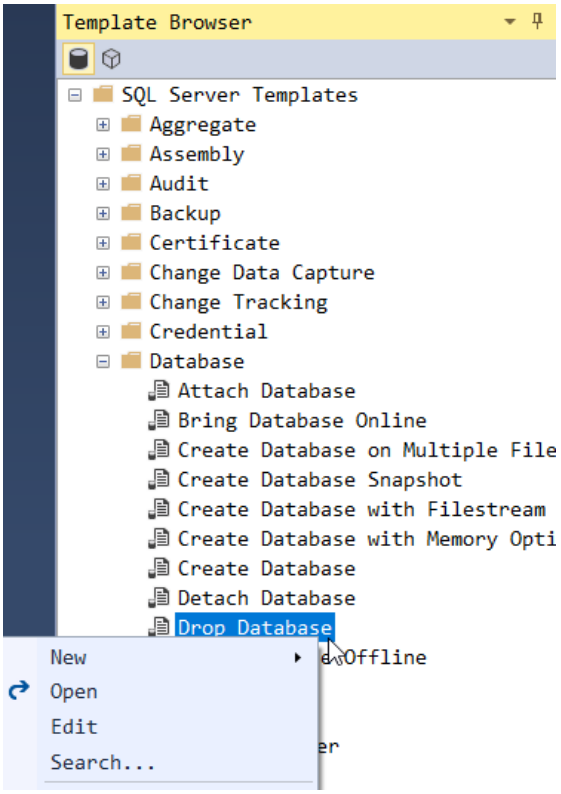
I thought I'd take dropping a database as an example. The supplied template doesn't work for me, so let's create a new one.

**Note:** SQL Server 2016 SP1 introduced **DROP DATABASE IF EXISTS** but I find that option quite useless. It fails if anyone is connected to the database. And to disconnect people beforehand, you need to first check if it exists, so the statement is pointless.

Let's start by opening the Template Explorer (from the View menu, click Template Explorer as it's not open by default).




Next we look for a DROP DATABASE option, and right-click it.



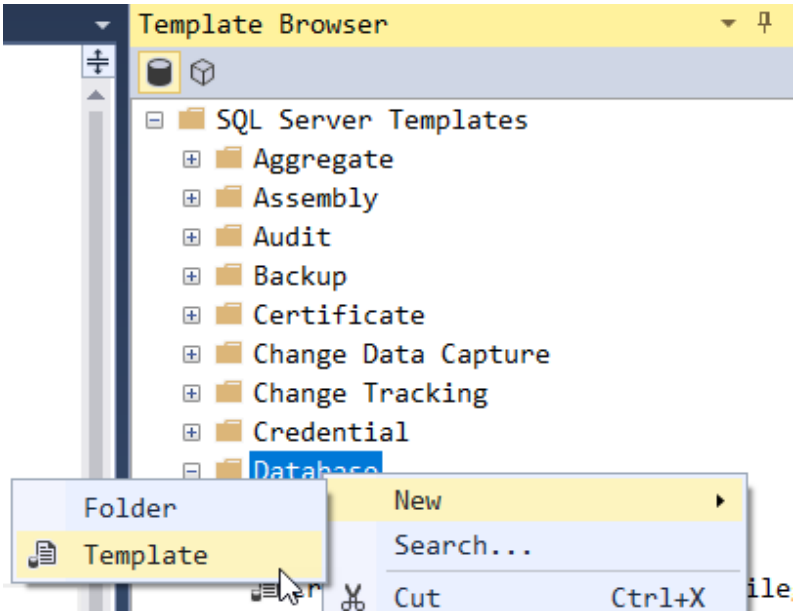
When we click Open, we see the supplied template:

```
-- ============================
-- Drop Database Template
-- ============================
USE master
GO

IF  EXISTS (
    SELECT name
        FROM sys.databases
        WHERE name = N'<Database_Name, sysname, Database_Name>'
)
DROP DATABASE <Database_Name, sysname, Database_Name>
GO
```

Importantly, note the text here that's shown between the less than and greater than operators. This is an example of a parameter to the template. Parameters look like:

**<ParameterName, DataType, DefaultValue>**

To create a new template, we could right-click the Database node in Template Explorer, and click New, then Template:




But to modify an existing one, it's easier to right-click the one we want to modify (ie: Drop Database), and click Copy. Then right-click where we want it (ie: the Database folder), and click Paste. That'll give us a copy of the template as a starting point.

Rename the new template to Drop Database - GL, right-click it, and click Edit.

I've changed the template as follows:

```
Drop Databas…0\Greg (129))*
USE master;
GO

IF EXISTS (SELECT d.[name]
           FROM sys.databases AS d
           WHERE d.[name] = N'<Database_Name, sysname, Database_Name>')
BEGIN
    ALTER DATABASE <Database_Name, sysname, Database_Name>
        SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE <Database_Name, sysname, Database_Name>;
END;
GO
```

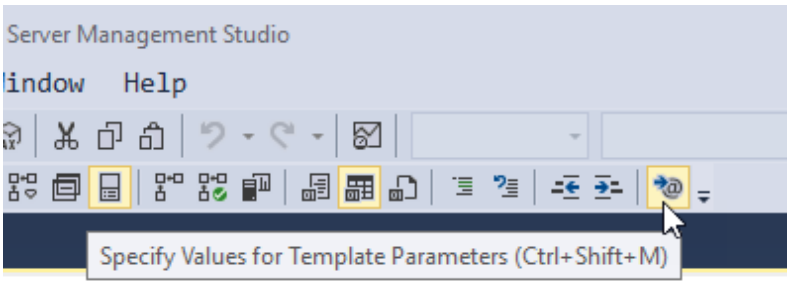
First up, I've used semicolons as statement terminators. We've been asked to use them since SQL Server 2005 so I have no idea why the existing templates (and much sample code from the SQL Server team) doesn't include them.

I've removed the comment headings. Otherwise, I'd need to do that every time. I've quoted the [name] column as I like to see the column names colorized properly, and this name would otherwise be blue. I've also aliased the table.

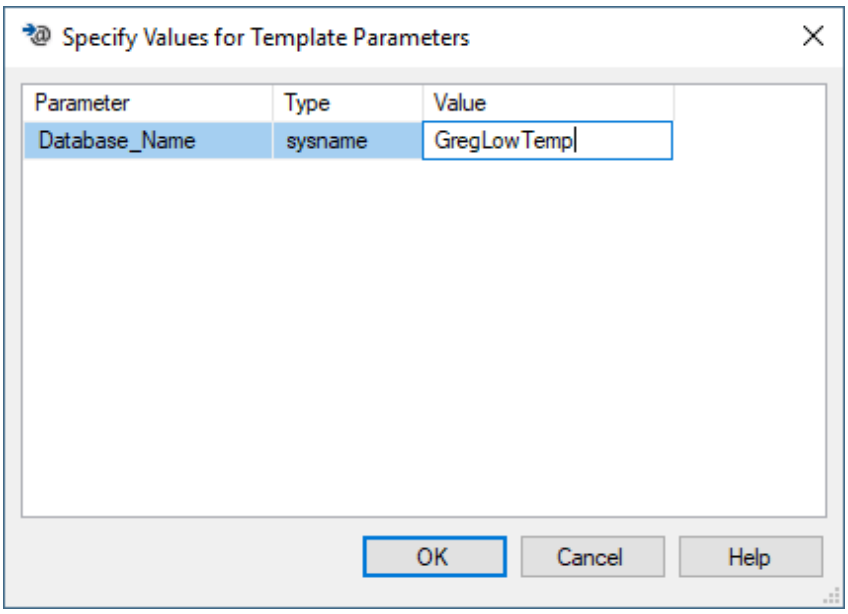
Next, to try to get around dropping a database with people already connected, I've altered it to single user, with an option to kick off all but a single user.

From the File menu, I click Save, then close the template window.

Now let's try it. From Template Explorer, I right-click **Drop Database - GL**, and click Open, and my new template appears. Then, on the toolbar, I click the option to let me enter the parameters:



I enter the parameter value:




After I click OK, my template is ready for use:

```
SQLQuery3.sq…0\Greg (131))*
USE master;
GO

IF EXISTS (SELECT d.[name]
           FROM sys.databases AS d
           WHERE d.[name] = N'GregLowTemp')
BEGIN
    ALTER DATABASE GregLowTemp
        SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE GregLowTemp;
END;
GO
```

The parameter value has been replaced wherever it had been used.

As a side note, this is still not 100% bulletproof. Note that the command is executed in the master database, and so there is no guarantee that the single user will be you. You could use RESTRICTED_USER but again, that might have the same issue. What is really needed is this:

```
DROP DATABASE IF EXISTS GregLowTemp WITH ROLLBACK IMMEDIATE;
```

But sadly, **that command doesn't exist**. There really should be a way to reliably drop a database by standard T-SQL. Many people need to write **drop and recreate** scripts.

Finally, I'll now point out that this code would be better as a snippet than as a template. Templates work well when they provide an entire script. This code is likely to be used as part of another script. In a later post, I'll show you how to make this a snippet instead.

## 3.14 Snippets in SQL Server Management Studio

Have you ever started to create an object using T-SQL in SQL Server, and though: "what's the right syntax for this?"

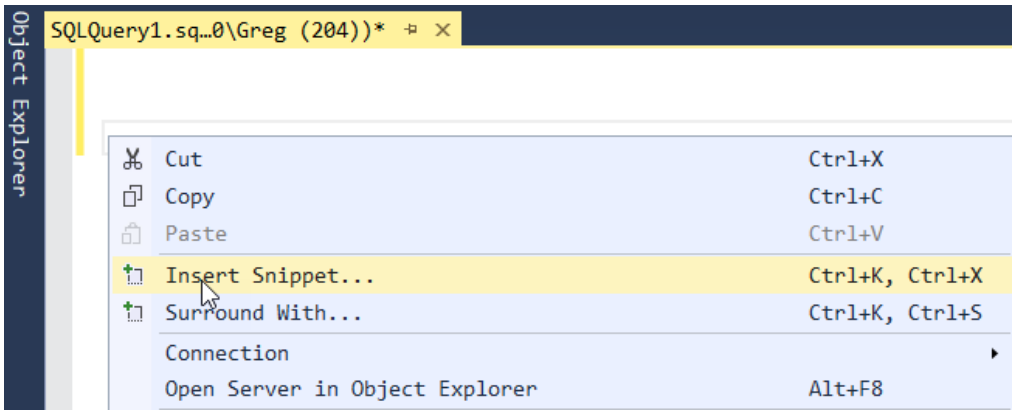
I've worked with SQL Server since 1992 (version 4.2) and yet almost every time I go to create a function, I must spend a few moments thinking about what the correct syntax is, because there are different types of functions (scalar vs table-valued, inline vs multi-statement).

SQL Server Management Studio has had templates for a long time, and they are useful. In fact, you can create your own.

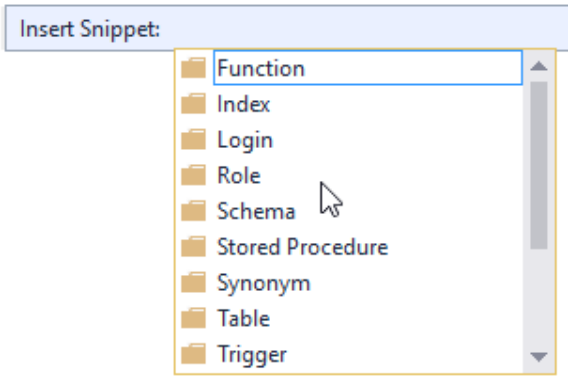
But what I wanted to focus on today, are the ones that were added a while back, yet most people seem to be unaware are there. And that's snippets.

At present, one of the easiest ways to get the basic syntax for creating an object is to use a snippet.

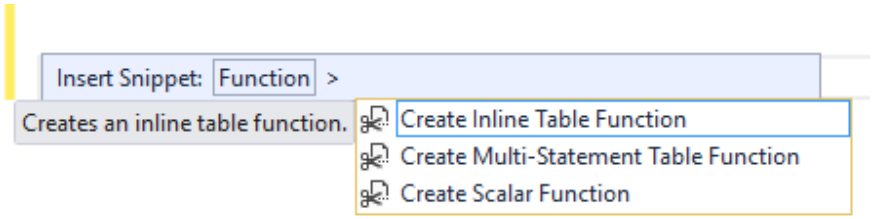
In a query window, right-click a blank area and note the option for inserting a snippet (you can also do this by Ctrl-K, then Ctrl-X):




Then select the type of object you want:



In this case, let's choose Function, and then we'll see the types of functions that are available:




Let's choose Create Inline Table Function, and SSMS fills in a skeleton for us:

```
CREATE FUNCTION [dbo].[FunctionName]
(
    @param1 int,
    @param2 char(5)
)
RETURNS TABLE AS RETURN
(
    SELECT @param1 AS c1,
           @param2 AS c2
)
```

Great. But don't get carried away with your mouse just yet. Notice that the template has placeholders that you can replace and the in this case, the schema is pre-highlighted. So I can just type the schema name, and hit tab, then type the function name, and hit tab, and so on until all placeholders are complete. Note that it has also stubbed out a SELECT statement for us, and as we change the parameter names, it changes the names in the SELECT as well.

```
CREATE FUNCTION [Sales].[GetTaxRatesByCategory]
(
    @ProductCategory int,
    @CountryCode nvarchar(10)
)
RETURNS TABLE AS RETURN
(
    SELECT @ProductCategory AS c1,
           @CountryCode AS c2
)
```

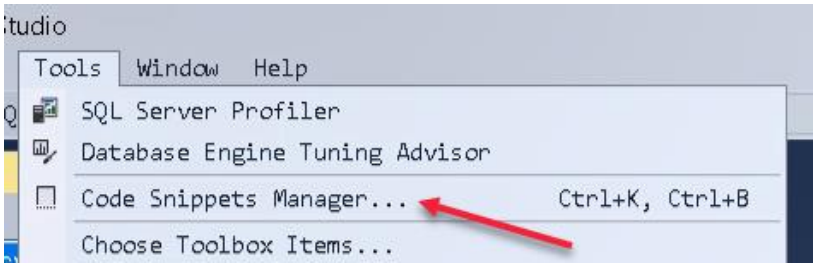
I've no doubt that the product team thinks I'm a bit pedantic about these things, but given we've been encouraging people to use semicolons as statement terminators since 2005, I wish they'd include that in the statement that they've provided. I must ping @sqltoolsguy about that. But regardless, snippets are wonderful and help to avoid that "what's the syntax for this" issue.

## 3.15 Using Snippets in SSMS to Improve the Drop Database Statement

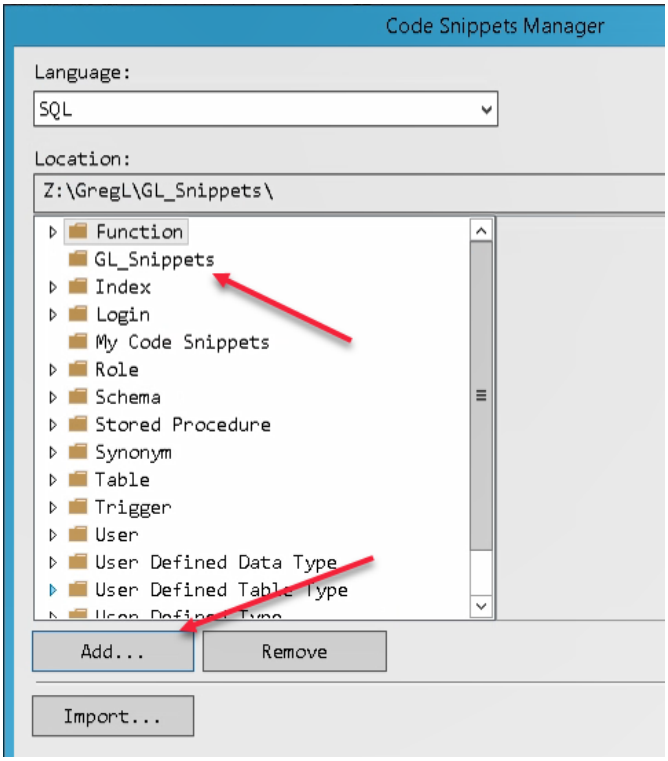
Earlier, I showed how to create a DROP DATABASE template in SQL Server Management Studio (SSMS). At the time, I mentioned that a template wasn't the best option because a command like this is normally inserted into a script; it's not the whole script.

That's where snippets shine. Let's create a snippet for it.

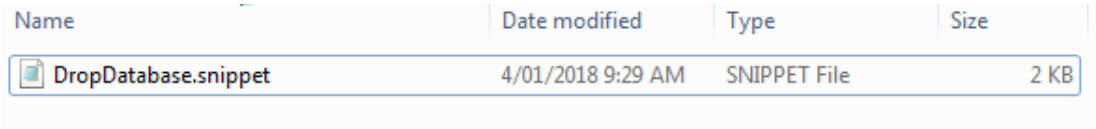
First let's open **Code Snippets Manager** (Tools > Code Snippets Manager):



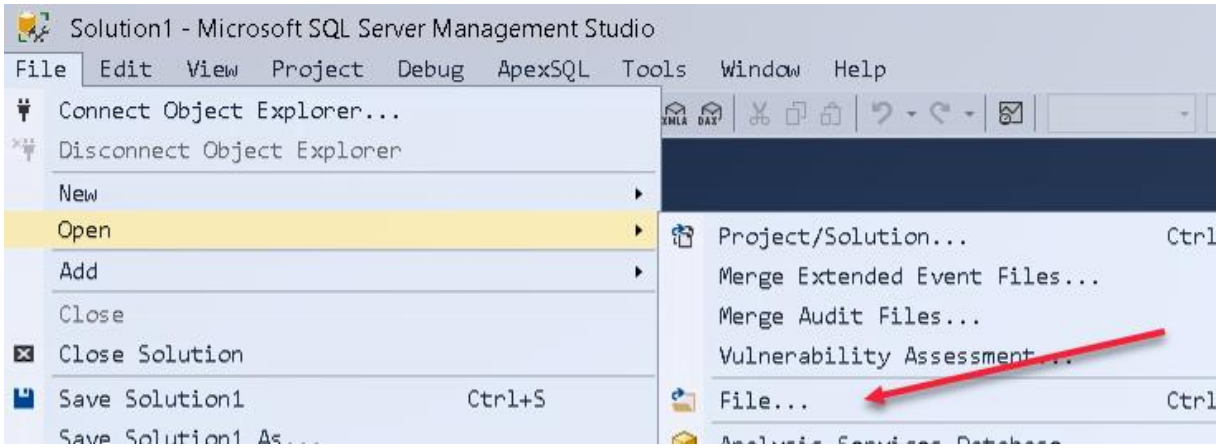

You'll see the existing snippet folders. I've clicked **Add**, then created a new folder called **GL_Snippets**.



Next I've created a file called **DropDatabase.snippet** on my desktop folder:




I then opened that file using SSMS:

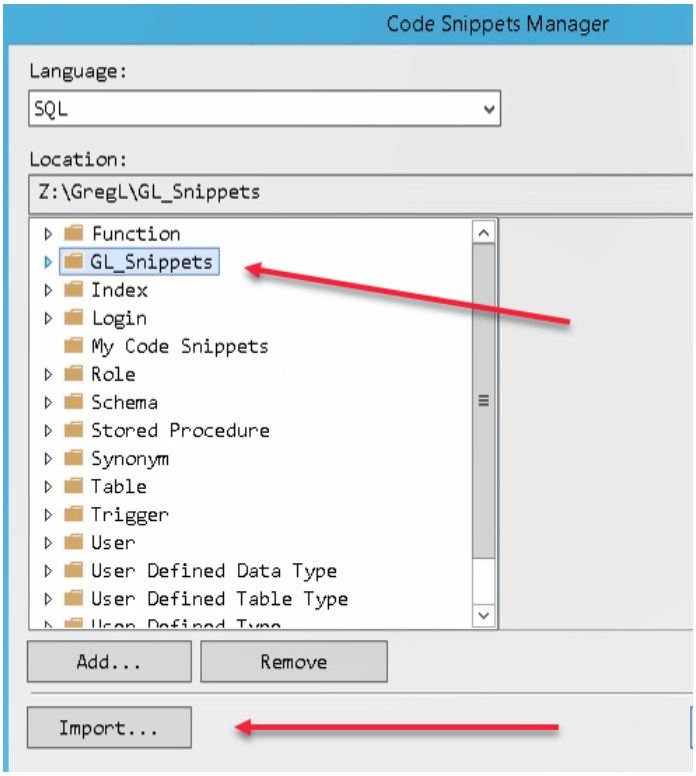



Note that SSMS has a perfectly good XML editor. I've then used it to create the snippet file and saved it:

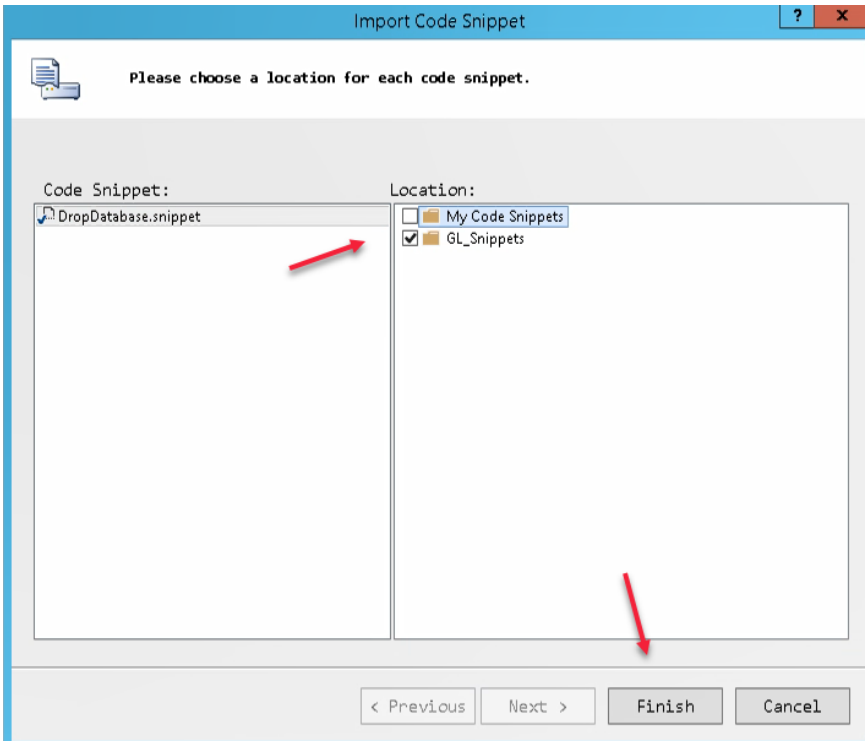
DropDatabase.snippet*

```
<?xml version="1.0" encoding="utf-8" ?>
<CodeSnippets  xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
    <CodeSnippet Format="1.0.0">
        <Header>
            <Title>DropDatabase</Title>
            <Description>Drop a database if it exists</Description>
            <Author>Dr Greg Low</Author>
        </Header>
        <Snippet>
            <Declarations>
                <Literal>
                    <ID>DatabaseName</ID>
                    <ToolTip>Name of the database to drop if it exists</ToolTip>
                    <Default>GregLowTemp</Default>
                </Literal>
            </Declarations>
            <Code Language="SQL"><![CDATA[
USE master;
GO

IF EXISTS (SELECT 1 FROM sys.databases AS d WHERE d.[name] = N'$DatabaseName$')
BEGIN
    ALTER DATABASE $DatabaseName$ SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE $DatabaseName$;
END;
GO
]]>
            </Code>
        </Snippet>
    </CodeSnippet>
</CodeSnippets>
```

Back in Code Snippets Manager, I've clicked **Import**:

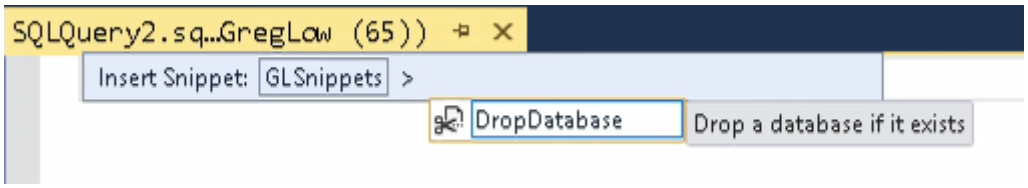



I located the file to import, deselected the **My Code Snippets** folder, and selected the **GL_Snippets** folder, then clicked **Finish**:



Now let's try the new snippet. I opened a new query window, and right-clicked in the empty space. We can then see the option for **Insert Snippet**(note we could have used Ctrl+K, Ctrl+X):


Then select our new DropDatabase snippet:




The snippet will then appear, with the literal parameter highlighted, ready for replacement:

```
USE master;
GO

IF EXISTS (SELECT 1 FROM sys.databases AS d WHERE d.[name] = N'GregLowTemp')
BEGIN
    ALTER DATABASE GregLowTemp SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE GregLowTemp;
END;
GO
```

I typed **SomeDatabase** and then clicked elsewhere, and it's all magically updated:

```
USE master;
GO

IF EXISTS (SELECT 1 FROM sys.databases AS d WHERE d.[name] = N'SomeDatabase')
BEGIN
    ALTER DATABASE SomeDatabase SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE SomeDatabase;
END;
GO
```

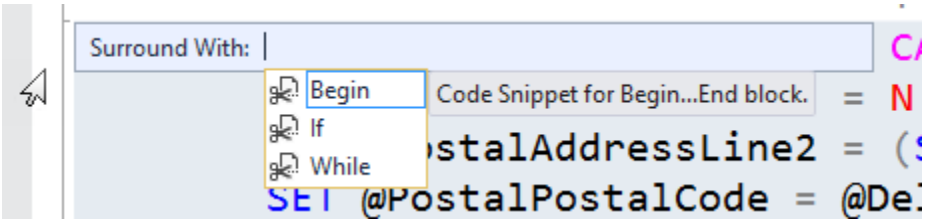
### 3.16 Using "Surround with" snippets

In another section, I've been talking about how to use snippets in SQL Server Management Studio (SSMS) and how to create your own. There are several types of snippets and one of the special types of snippets that I want to mention are the "surround with" snippets.

If you look at the following block of code:

```
        SET @DeliveryAddressLine2 = CAST(CEILING(RAND() * 2000) + 1 AS nvarcha
                                  + (SELECT TOP(1) PreferredName FROM [Applica
                                  + N' ' + @StreetSuffix;
        SET @DeliveryPostalCode = CAST(CEILING(RAND() * 800) + 90000 AS nvarch
        SET @PostalAddressLine1 = N'PO Box ' + CAST(CEILING(RAND() * 10000) +
        SET @PostalAddressLine2 = (SELECT TOP(1) PreferredName FROM [Applicati
        SET @PostalPostalCode = @DeliveryPostalCode;

        SET @PrimaryContactPersonID = NEXT VALUE FOR Sequences.PersonID;
```

Imagine that you want to execute the four highlighted lines only when a condition is true. If I hit Ctrl-K and Ctrl-S while they are highlighted, I'm prompted with this:



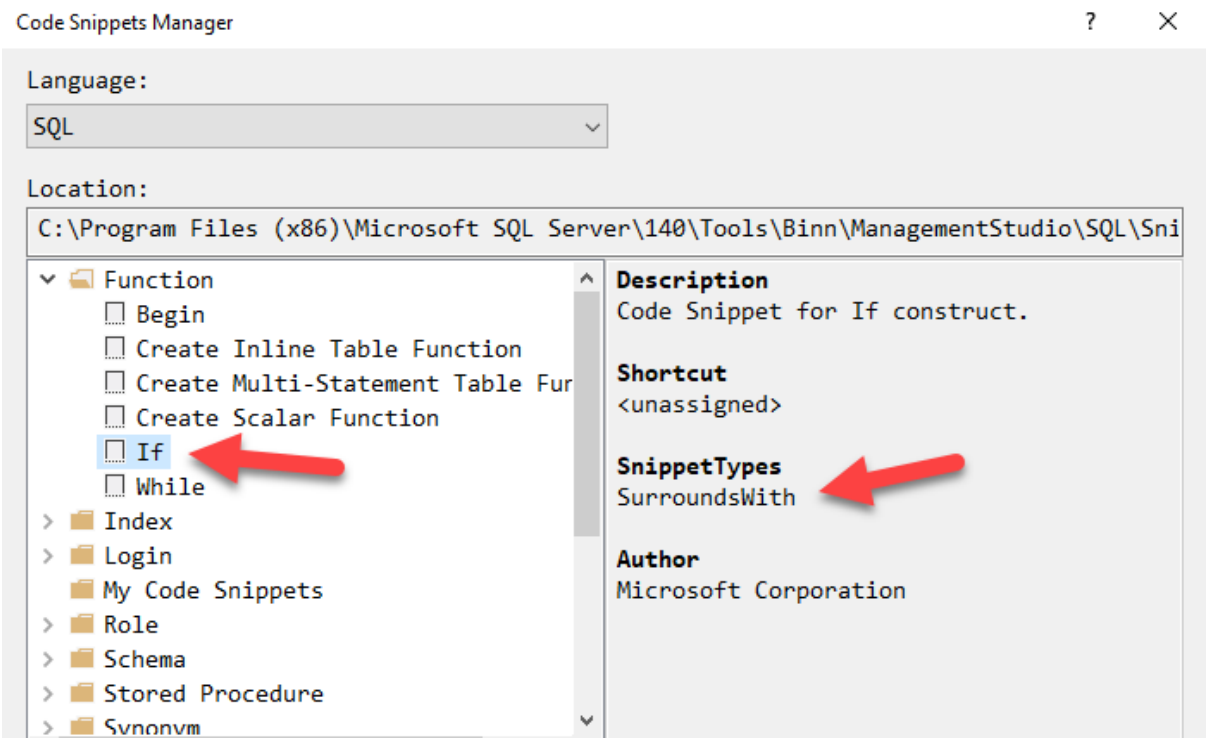

Note that I get an option to surround them with a BEGIN/END, or an IF or WHILE. Let's choose an IF. I just double-click the If option and this happens:

```
        SET @DeliveryAddressLine2 = CAST(CEILING(RAND() * 200
                                  + (SELECT TOP(1) PreferredN
                                  + N' ' + @StreetSuffix;


IF( Condition )
BEGIN

        SET @DeliveryPostalCode = CAST(CEILING(RAND() * 800)
        SET @PostalAddressLine1 = N'PO Box ' + CAST(CEILING(R
        SET @PostalAddressLine2 = (SELECT TOP(1) PreferredNam
        SET @PostalPostalCode = @DeliveryPostalCode;


END
```

We can then just type the condition. Notice that because I was using multiple lines, it put them all in a BEGIN/END block for me.

This is all quite good but I still might want to create my own instead, and I can do that. An example of why I might want to do that, is that I might want a statement terminator after the END.

If we go into the Code Snippets Manager (from the Tools menu), and expand the Function category, we can see this:




I can't say that I think of IF as a function but none-the-less, note that the type is a SurroundsWith snippet. Let's see how it's defined. The Location is here:

**C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Binn\ManagementStudio\SQL\Snippets\1033\Function\If.snippet**

If I open that file in NotePad++, I see the following. Note how the value called **$selected$** is enclosed within the snippet.

```
<?xml version="1.0" encoding="utf-8" ?>
<CodeSnippets  xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
<_locDefinition xmlns="urn:locstudio">
    <_locDefault _loc="locNone" />
    <_locTag _loc="locData">Title</_locTag>
    <_locTag _loc="locData">Description</_locTag>
    <_locTag _loc="locData">Author</_locTag>
    <_locTag _loc="locData">ToolTip</_locTag>
   <_locTag _loc="locData">Default</_locTag>
</_locDefinition>
    <CodeSnippet Format="1.0.0">
        <Header>
            <Title>If</Title>
                    <Shortcut></Shortcut>
            <Description>Code Snippet for If construct.</Description>
            <Author>Microsoft Corporation</Author>
            <SnippetTypes>
                        <SnippetType>SurroundsWith</SnippetType>
            </SnippetTypes>
        </Header>
        <Snippet>
            <Declarations>
                        <Literal>
                            <ID>Condition</ID>
                            <ToolTip>Condition to evaluate</ToolTip>
                            <Default>Condition</Default>
                        </Literal>
            </Declarations>
            <Code Language="SQL"><![CDATA[
IF( $Condition$ )
BEGIN

$selected$ $end$

END
]]>
            </Code>
        </Snippet>
    </CodeSnippet>
</CodeSnippets>
```

I could then either modify this one (probably not best), or use it as the basis of a new snippet for myself.

# 4 Executing Queries

## 4.1 Adding additional parameters to connections

When I am writing my own code using a .NET (or other) language, I have a great deal of control of how the connection string that my application uses to connect to SQL Server is configured.

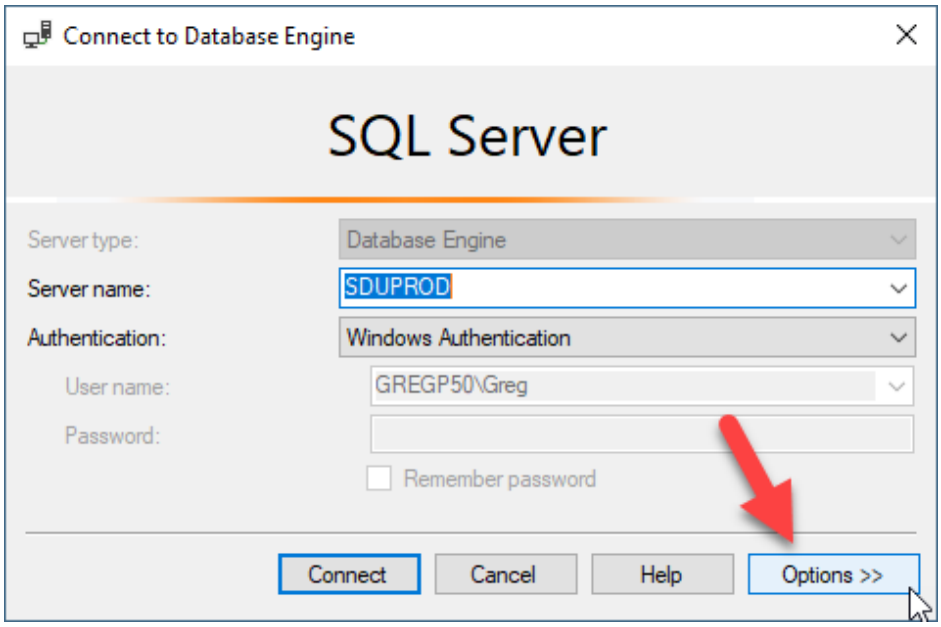
In particular, I might need to add another parameter or two.

As a simple example, you might have a multi-subnet Availability Group, spread across a production site and a disaster recovery site. It's common to then have an Availability Group Listener in both subnets.

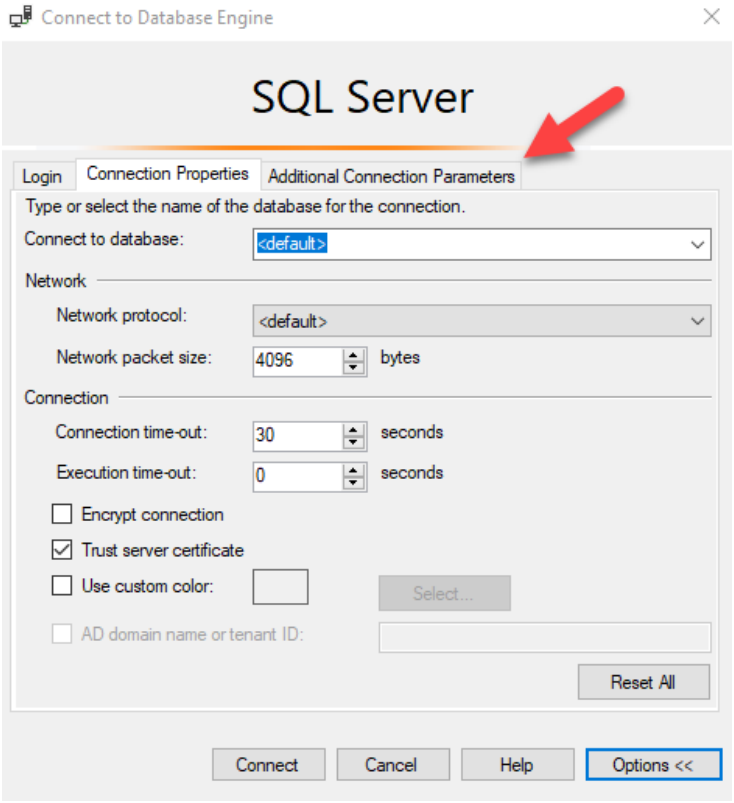
If you add the parameter **MultiSubnetFailover=true** to your connection string, when SQL Server attempts to connect to the listener, it will send a request to each IP address concurrently, not just to one at a time. It will then connect to whichever server responds first.

This is great, but how do we do that with SQL Server Management Studio (SSMS) connections?

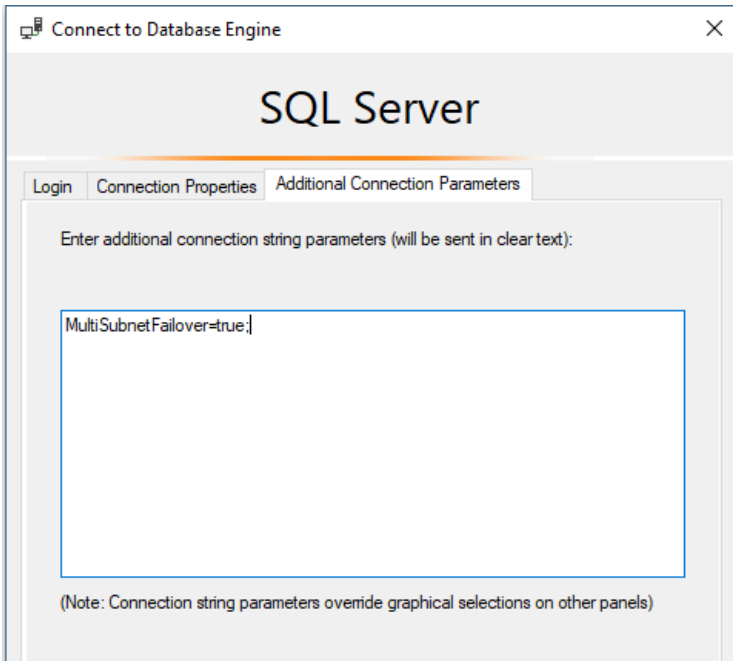
The answer is that in the database server connection dialog, we can choose Options:



In the dialog that appears, there is a tab for Additional Connection Parameters:




On that tab, I can enter the required details:




Note also that if you enter a value here that is also on the graphical connection pages, the value that you enter overrides those values.

## 4.2 Using Colors to Avoid Running Scripts Against the Wrong Server

Everyone who's worked with SQL Server for any length of time, has had the experience of executing a T-SQL script, and then noticing, with horror, that they've just executed the script against the wrong server.

You know the feeling. It even happens at Christmas time, just when you were hoping to get away from work for a few days, or when you are the unlucky one who's doing on call work.

Many of these surprises would be avoided if there was something that gave you a visual clue that you were connected to the wrong server.

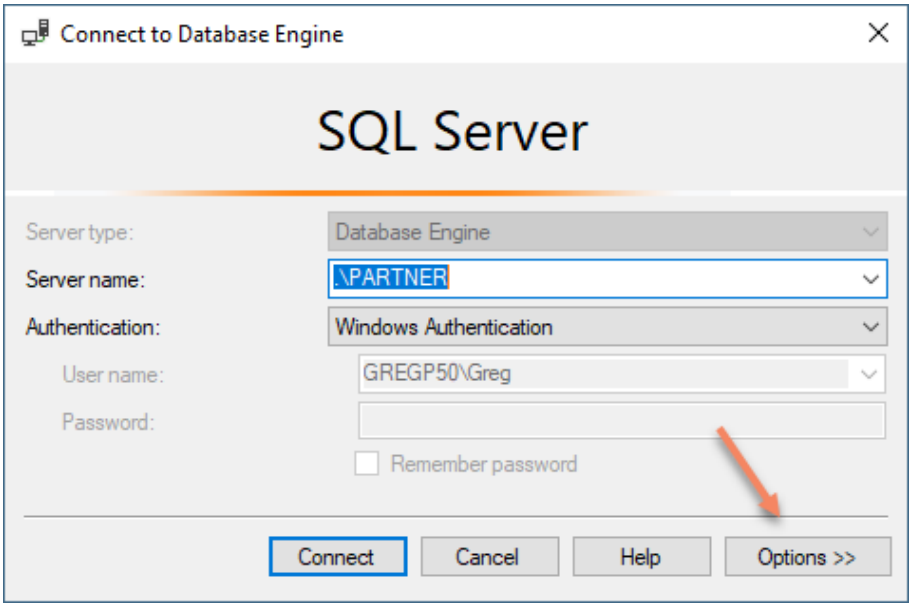
SQL Server Management Studio (SSMS) has had an option to color code connections to servers for quite a while. The solution isn't perfect and isn't as good as Mladen Prajdić's SSMS Tools Pack:

http://www.ssmstoolspack.com/

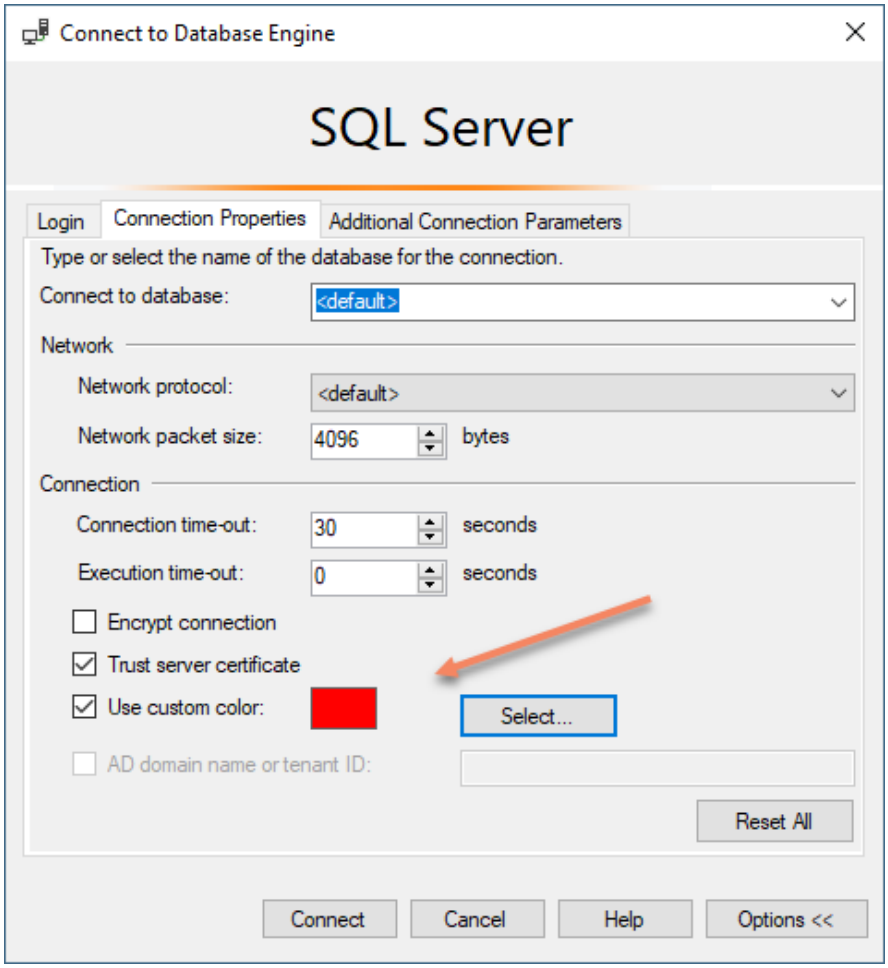
(If you're keen to pay for an add-in, I do recommend that one)

For many people though, the colorizing provided by SSMS is just enough. And it's easy to use.

When you open a new database connection, you can click **Options**:

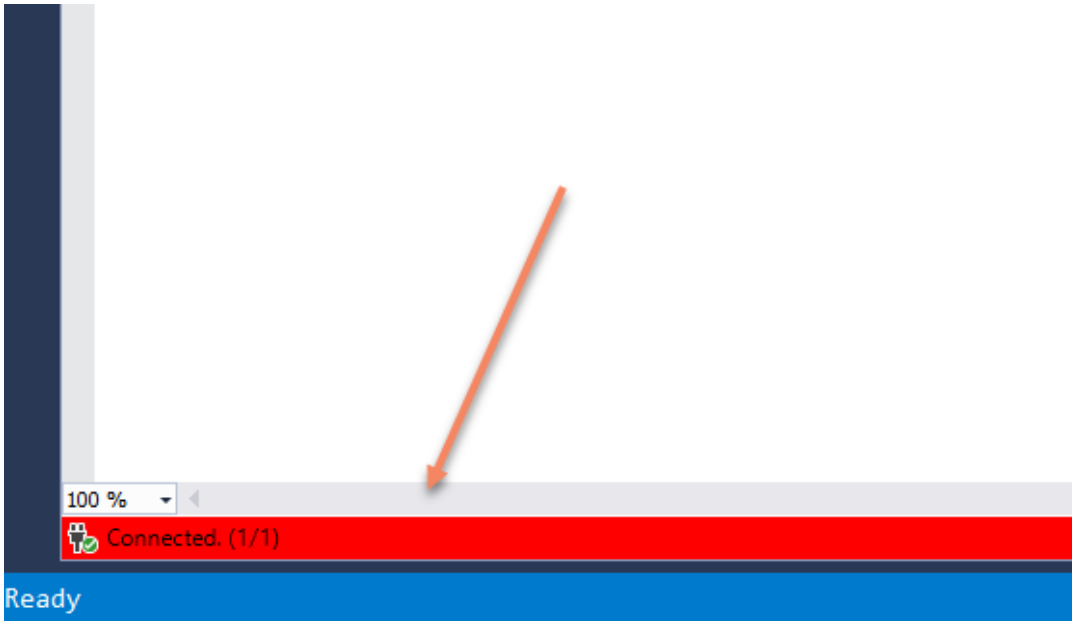



This opens a larger dialog that allows you to specify a color for the connection:



I've chosen Red to warn me that this is a production server.

Then, you'll see the color bar at the bottom of each script window in SSMS:



### 4.3 Change connection

I commonly run into a few connection-related scenarios:

- I'm working on a large query and need to run it against several servers, not concurrently, but one after the other.
- I've just made a database connection, and got my query ready, only to discover that I've connected to the wrong server.

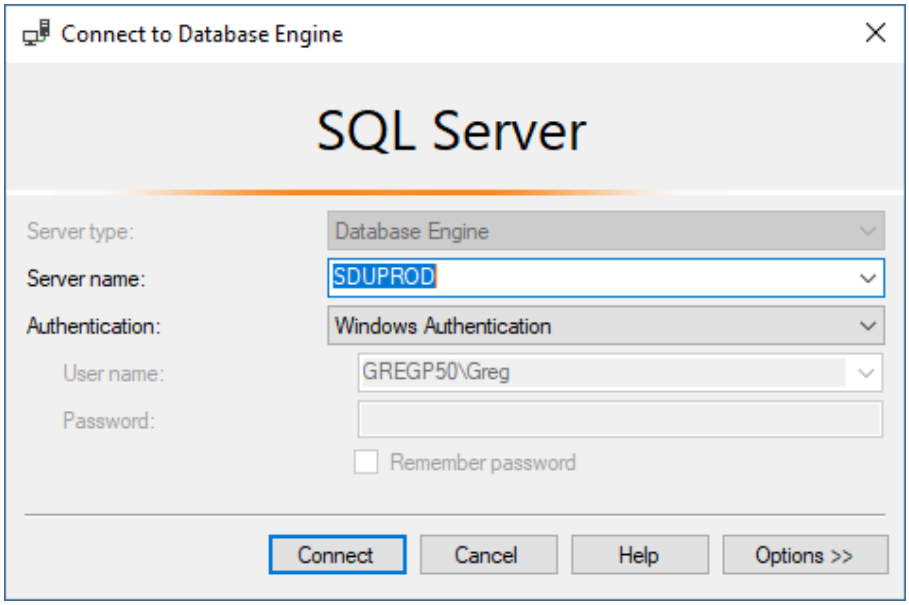
Either way, what I've seen people do in these scenarios is to:

- Select all the text in the current query
- Copy it
- Open a new query window
- Paste the code

That's all good but SQL Server Management Studio (SSMS) has had a simpler way of doing this for quite a while.

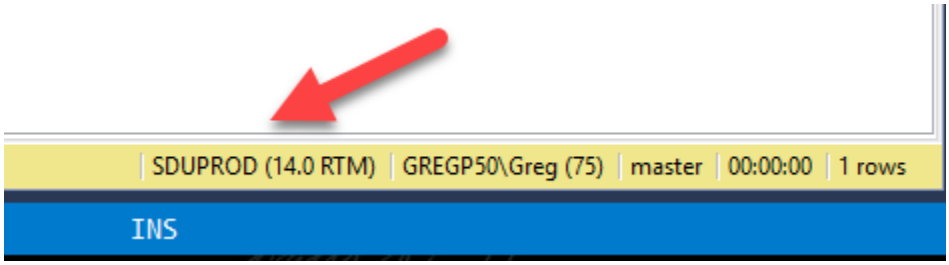
If you right-click in the middle of a query window, you can choose to change the connection. Here's an example. I opened a new query window and entered a query, only to find that it connected to the Azure server that I was working with earlier. The easiest option is to right-click and choose to change the connection:


After you click **Change Connection**, you can log on again:

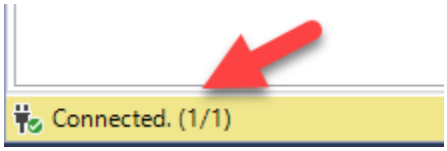



And then I can continue in the same query window as before. This is particularly useful if I need to run some code against a test server, and once I've decided that it was correct, I can just change the connection and connect to the "real" server.

At the bottom-right of your query window, you can always see which server you are connected to:




And at the bottom-left, you can see your connection state:

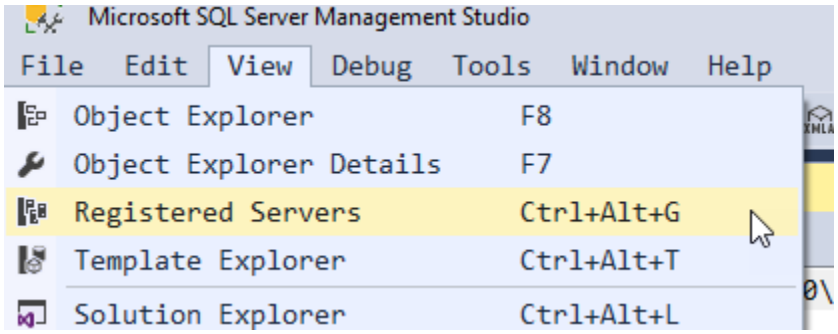


## 4.4 Configuring registered servers

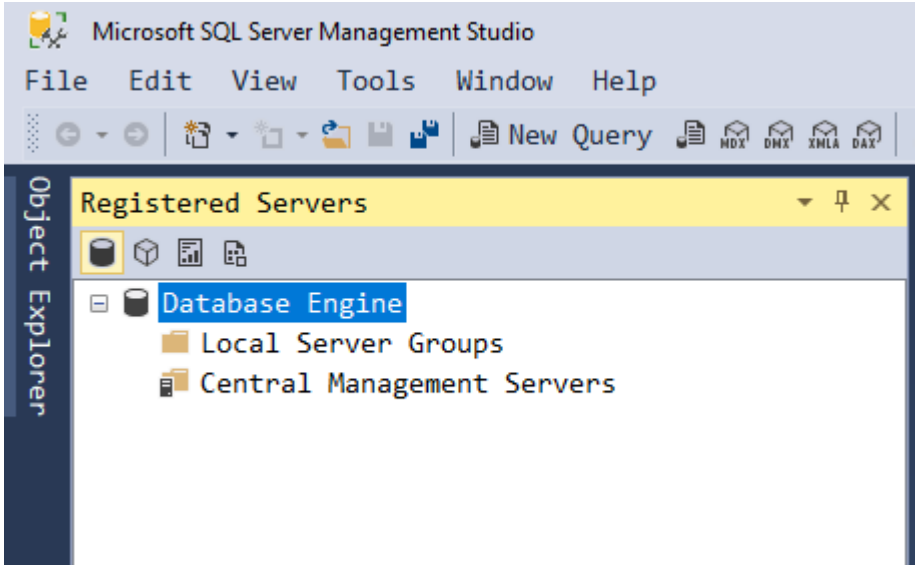
When working with SQL Server systems, it can be hard to remember the names of all the servers, to remember connection details for the ones that need SQL logins (instead of Windows authentication), and to remember other details of those servers, such as which environments they are part of (eg: production, UAT, test)

SQL Server Management Studio (SSMS) has a facility to help you to do this. It allows you to register server details in a single place.

By default, the window isn't shown, but from the **View** menu, you can choose **Registered Servers**.



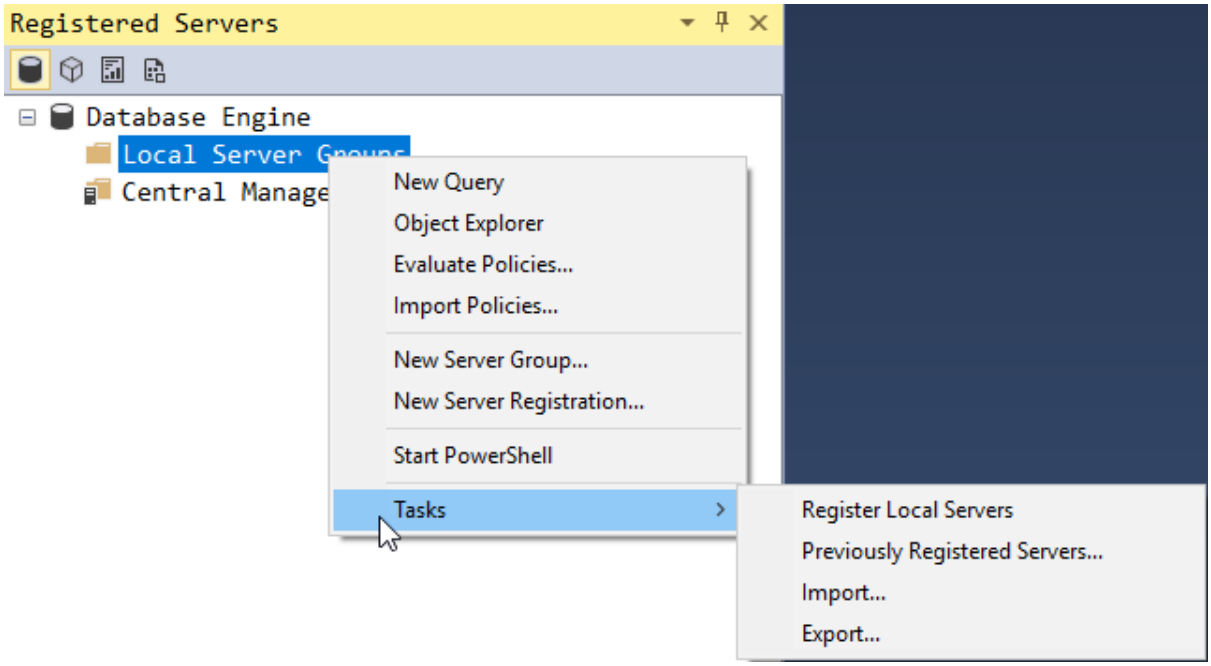

When the window opens, you can see this:




Note the toolbar near the top of the window. It is showing that we're configuring database servers but the other icons let you know that you can also work with Analysis Services, Integration Services, and Reporting Services servers.

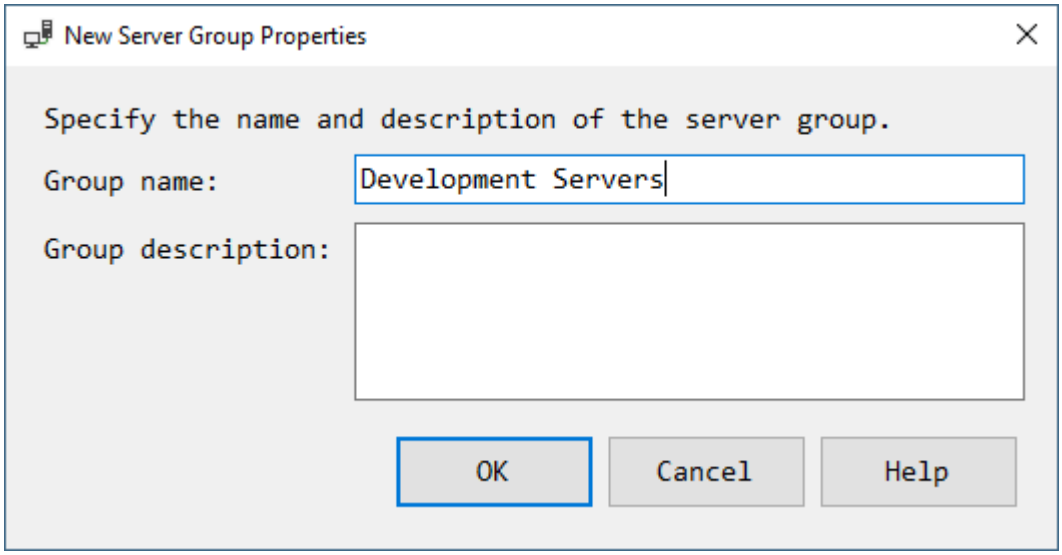
The first decision that you need to take is to decide where the details will be stored. **Local Server Groups** are stored on your local system ie: the system that is running SSMS. If you move to a different system to work, you won't have those connection details. Alternately, a Central Management Server can be configured. This is a server that agrees to hold connection details. While this seems a great idea (because the details would be held in a single place), one of the down-sides of this arrangement is that only Windows authentication can then be used. Local Server Groups can also work with SQL logins.

Let's create a server group as an example. If I right-click Local Server Groups, here are the available options:

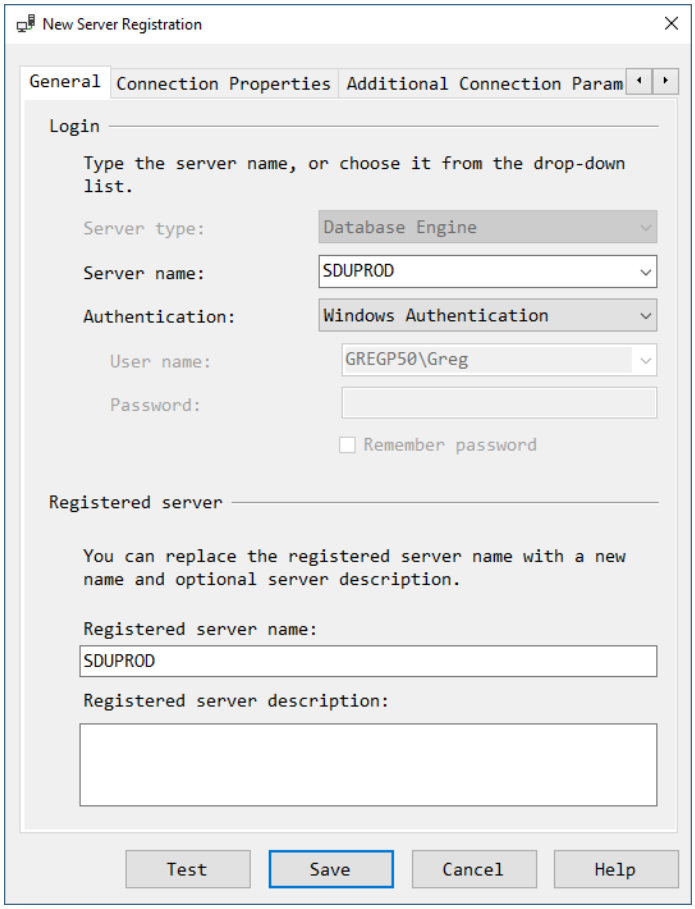



Note that there is an option to Import (and Export) these details. This at least allows you to move details between systems.

Let's create a new Server Group:



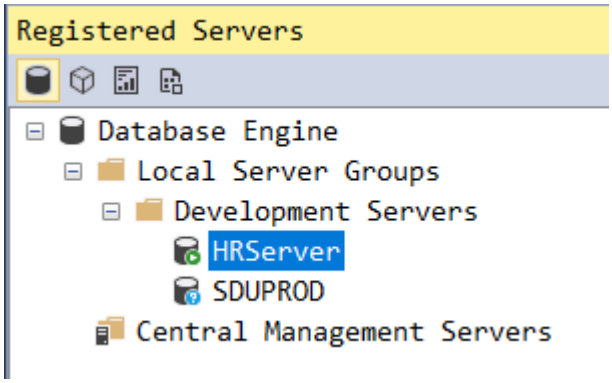
It just needs a name and an optional description, then OK. When it's created, right-click it, and choose New Server Registration:




I've connected to the server SDUPROD and I've given the registered server the same name. Note that you don't need to do that. I could have called it PayrollServer or some other more meaningful name. You'll also notice that there are tabs for configuring other connection properties.

I've then created a second server called HRServer and under the covers, I've pointed it to another server.

Now I have all my servers in groups, in an appropriate location. I can right-click them to open new queries to them, and to do much more.

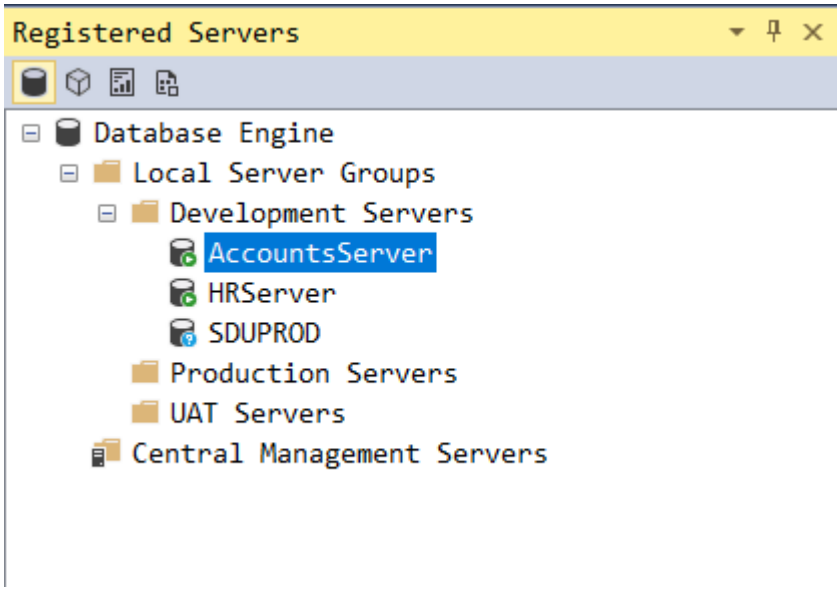


### 4.5 Multi-server Queries

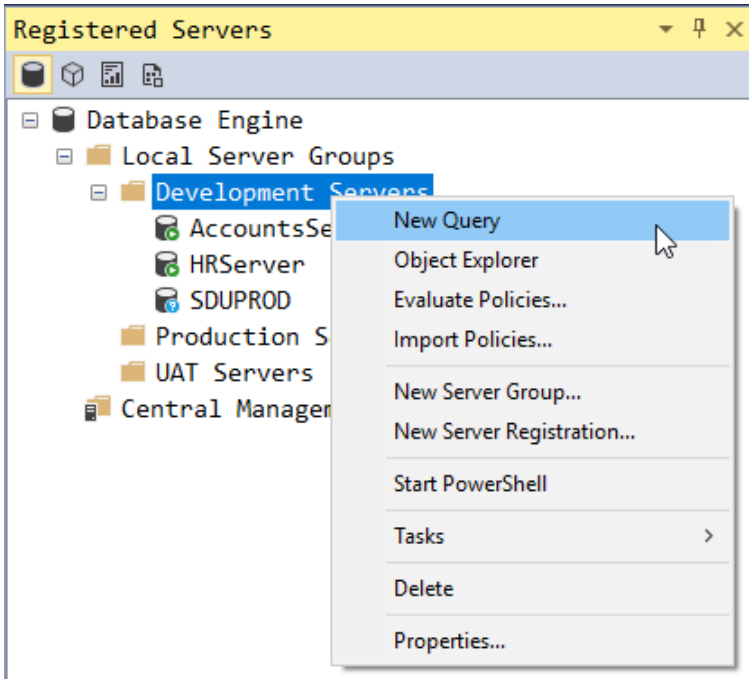
In an earlier post, I mentioned that you can create a registered list of servers, either in Local Server Groups or stored in a Central Management Server.

What I didn't really talk about though, is what you can do with these groups of servers, rather than just executing queries on an individual server.

I've created three local server groups, for my development, UAT, and production servers.



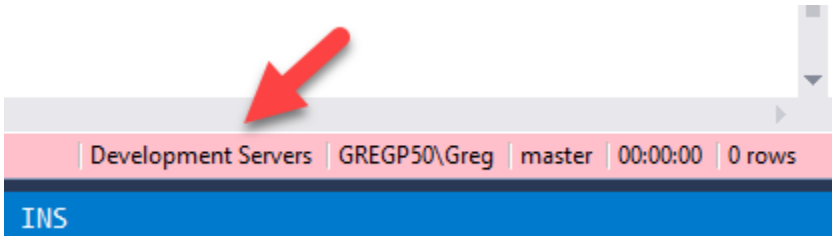

The Development Servers group has three database servers in it. If I right-click the group, rather than any individual server, we get these options:




I'll talk about policies another day but notice that you can import or check (evaluate) policies across a whole group of servers.

But the option that interests me today is the **New Query** option. When you click this, it opens a query window for the group of servers.

The window color at the bottom has changed from the default, and in the bottom right, we can see that the window is connected to the local server group:




If I type the query **SELECT @@VERSION;** and click **Execute**, I see this:

```
SELECT @@VERSION;
```

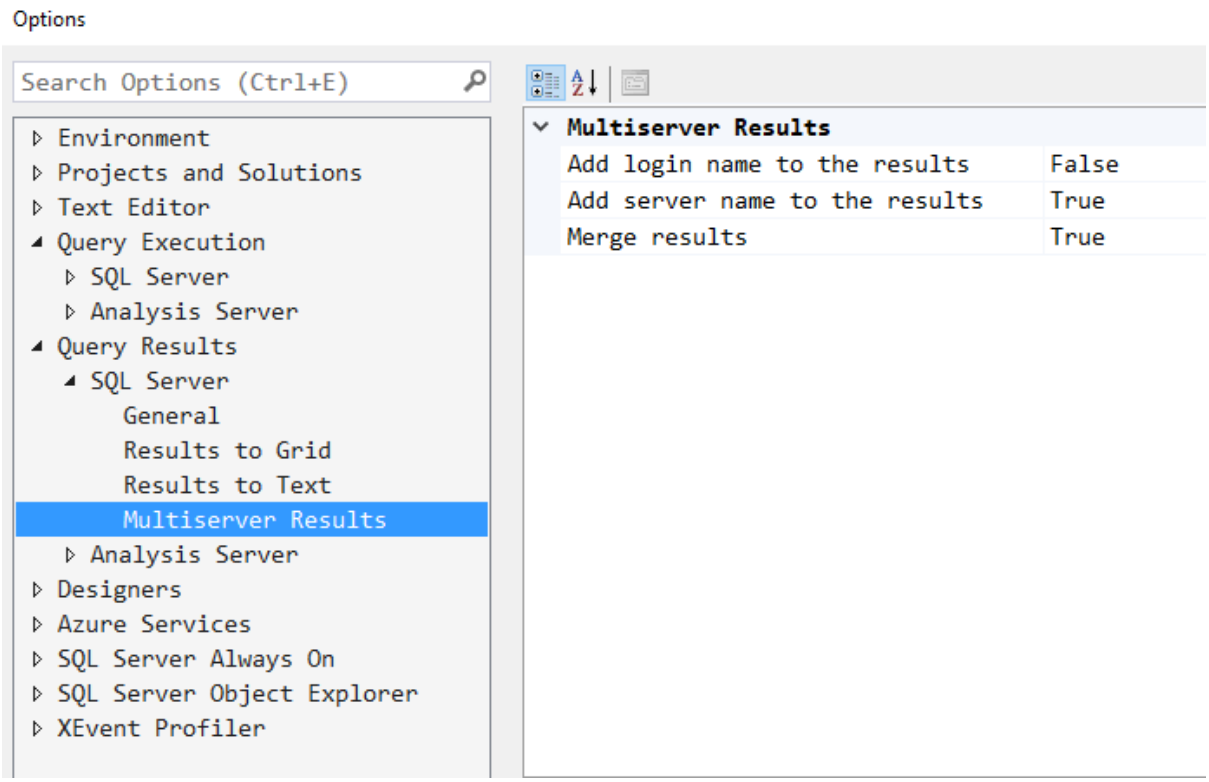
100 %

Results | Messages

| | Server Name | (No column name) |
|---|---|---|
| 1 | HRServer | Microsoft SQL Server 2017 (RTM-CU6) (KB4101464... |
| 2 | AccountsServer | Microsoft SQL Server 2017 (RTM-CU6) (KB4101464... |
| 3 | SDUPROD | Microsoft SQL Server 2017 (RTM-CU6) (KB4101464... |

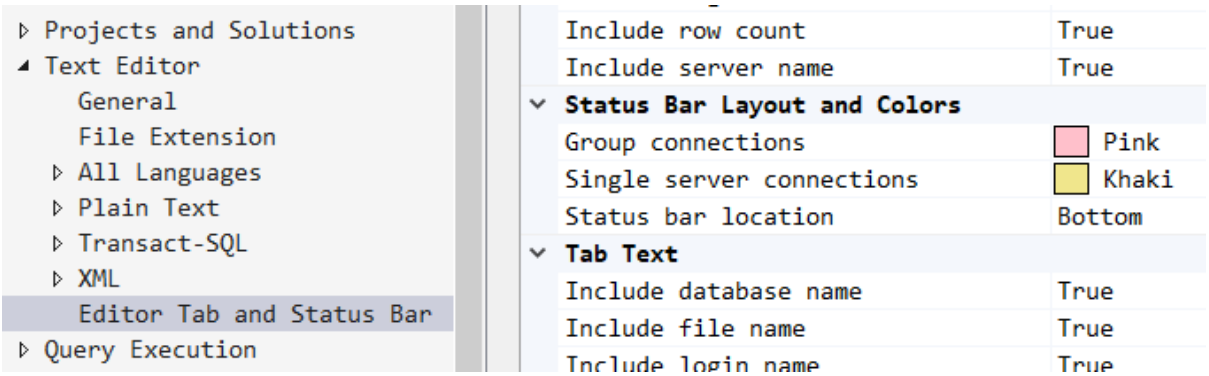
Even though this looks like a single result set, this is just a SQL Server Management Studio (SSMS) trick. Under the covers, it has run the query individually against each server. It has just presented the results to us as though they are a single set of results.

We can configure several things about how this happens. In Tools, Options, Query Results, SQL Server, Multi-server Results, we have these options:




We could add a column that shows our login name for each server to the results. We could remove the server name if required, although that doesn't seem very useful. And we could choose to not have the results merged. If we do that, SSMS returns a separate result set for each server.

You can also change the color of the bar below, in this location:

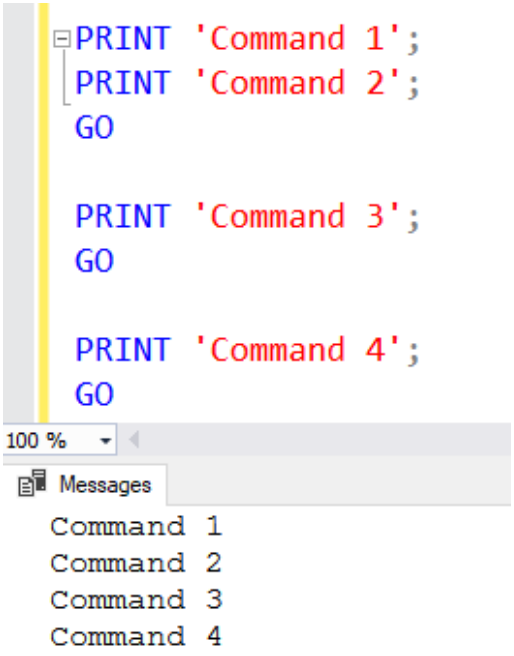



Multi-server queries were an interesting addition to SSMS. They are useful for a relatively small number of servers. As the number of servers increases, they would become more fragile, and you might want to consider using a 3rd party tool that works out which servers have or haven't had the query run, retry options, etc.

### 4.6 Using a Count with the GO Batch Separator in T-SQL

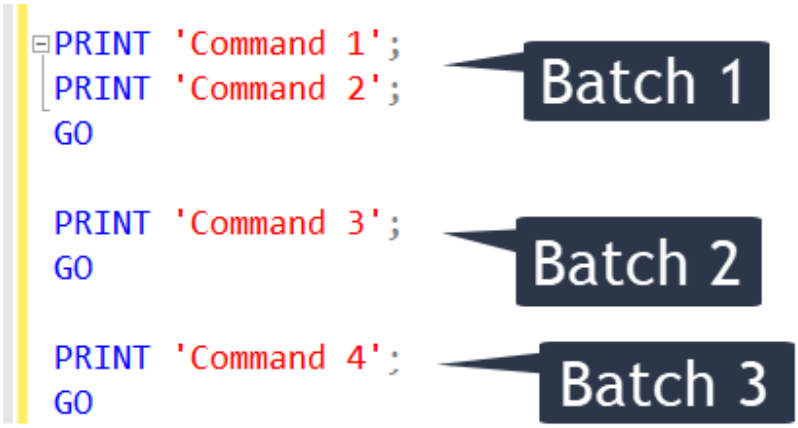
In T-SQL, a script is a set of one or more batches.

For example, if we have the following script and click Execute, it looks like all the commands were sent to the server and executed all at once:




But that isn't what happened.

What did happen is that SQL Server Management Studio (SSMS) found the word **GO** and broke the script into a series of batches. In this case, there were three batches. First, it sent the commands shown here as Batch 1 to the server, waited for them to execute, then sent Batch 2, waited for it to execute, then sent Batch 3, and waited for it to execute.

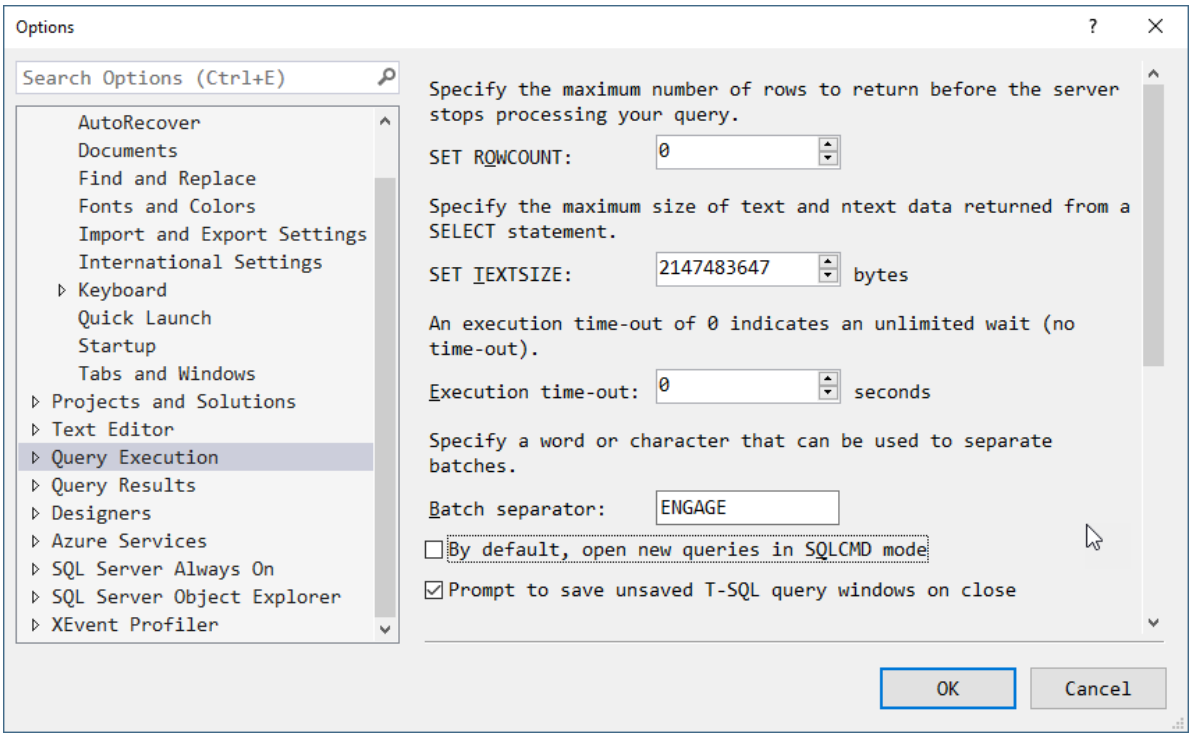



It looks like it just happened all at once, but it didn't.

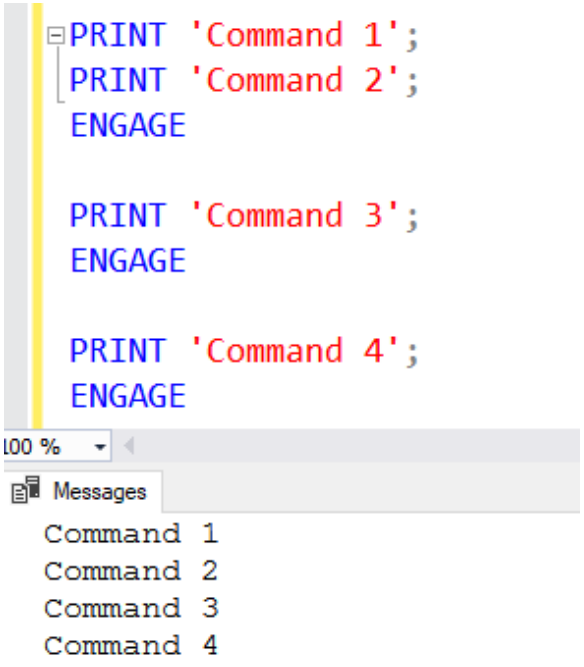
The key thing to understand about GO is that it's not a T-SQL command. It's a batch separator. We use GO to separate batches of commands that are part of a single script. The word GO is never sent to SQL Server. It only has meaning to the client tool (ie in this case SSMS).

In fact, you can change it. One of my friends that's a Star Trek fan has all his scripts with **ENGAGE** rather than **GO**.

You can see that I've changed it in Tools -> Options :



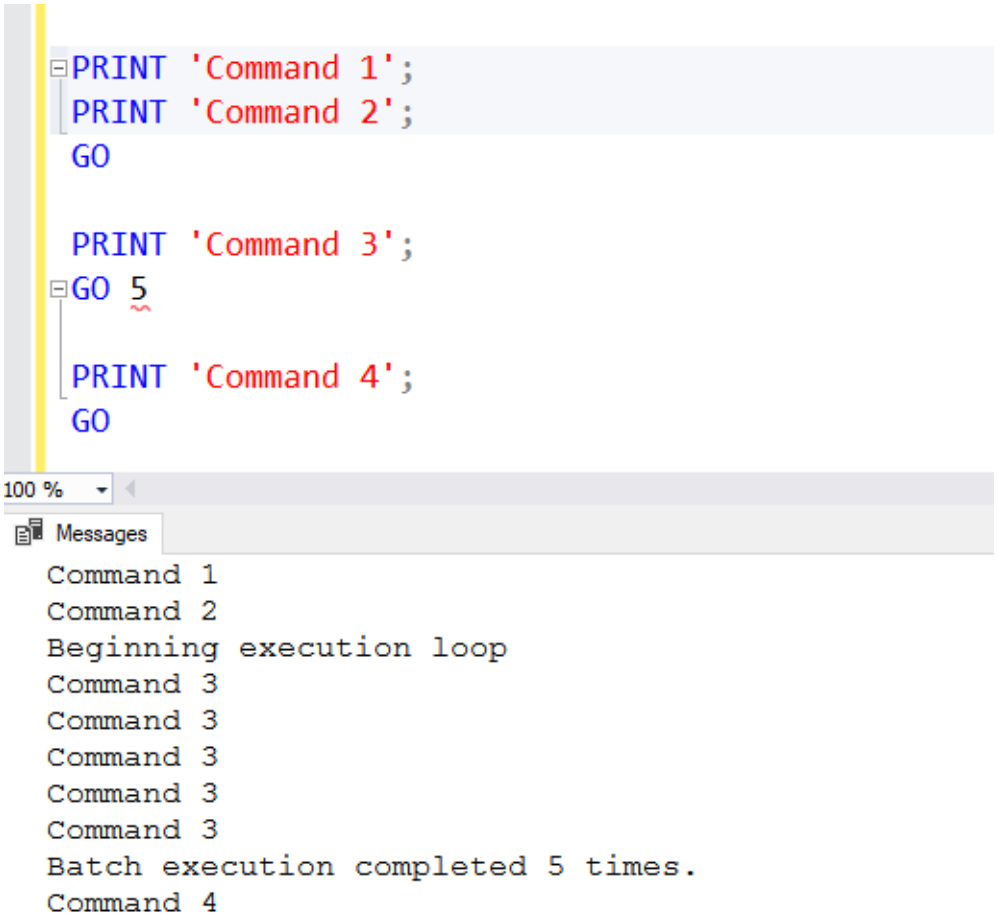

And now if I open a new query window, that works just fine:




I can't recommend doing that as your scripts won't be much use to anyone else.

So keep in mind that it's the client that understands GO, not the server. And that leads us to our shortcut of the day.

You can add a count after the word GO, and then SSMS will send your commands from that batch to the server more than once:




Notice that Intellisense in SSMS doesn't understand the syntax (there is a red squiggle) but it works just fine.

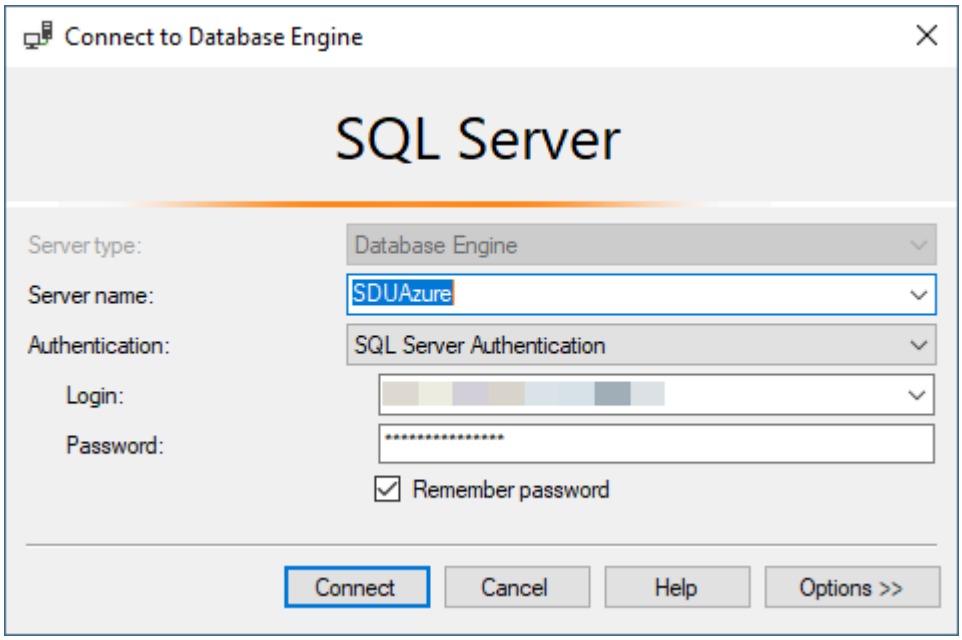
I find this shortcut quite useful if I want to execute a command a substantial number of times. An example is when I want to insert a bunch of default rows into a table; another is for executing commands while testing.

## 4.7 Viewing Client statistics

While SQL Server is quite fast at executing queries, when you are connecting from a client application like SQL Server Management Studio (SSMS), you might wonder how much time SQL Server spent executing the query, as opposed to how long the communication with the server took.

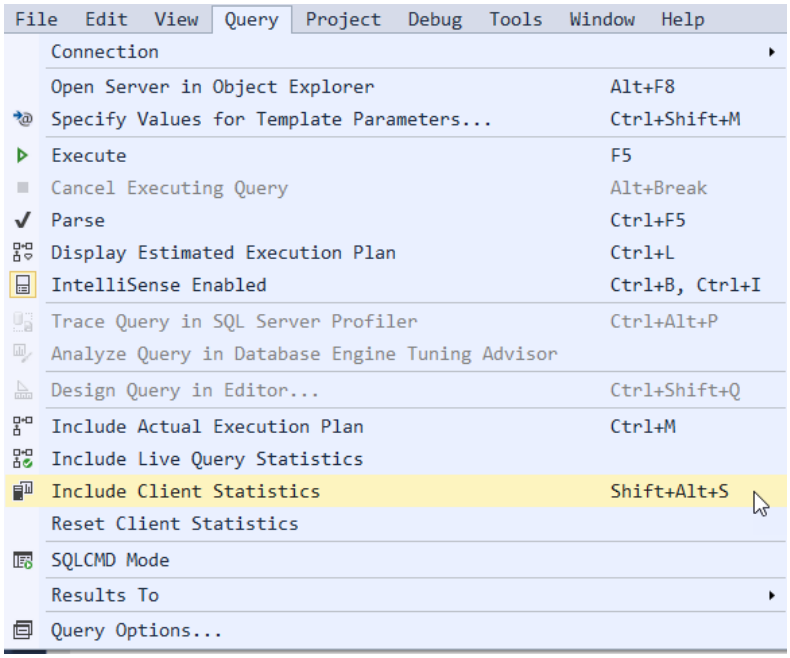
This type of information is available in the **Client Statistics**.

Let's see an example. If I connect to a server in an Azure data center, I'll have higher latency than for one in my own site. That will affect the wait time for a server response.



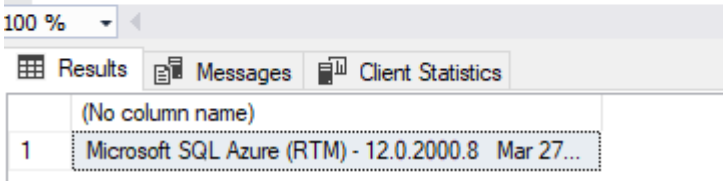

This server is in the Melbourne (Australia South East) data center.

Let's execute a simple query against it, but before doing so, on the **Query** menu, choose **Include Client Statistics**.

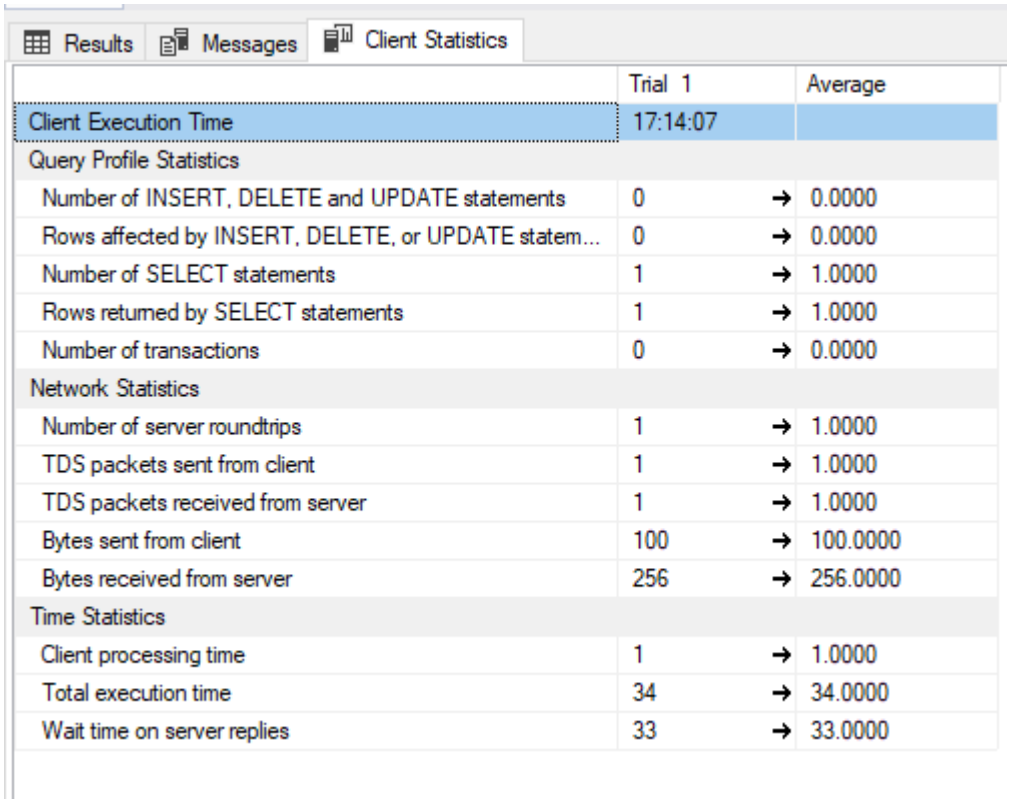


I'll just query the SQL Server version:

```
SELECT @@VERSION;
```




Notice that an extra tab of data is returned.




| | Trial 1 | Average |
|---|---|---|
| Client Execution Time | 17:14:07 | |
| Query Profile Statistics | | |
| Number of INSERT, DELETE and UPDATE statements | 0 → | 0.0000 |
| Rows affected by INSERT, DELETE, or UPDATE statem... | 0 → | 0.0000 |
| Number of SELECT statements | 1 → | 1.0000 |
| Rows returned by SELECT statements | 1 → | 1.0000 |
| Number of transactions | 0 → | 0.0000 |
| Network Statistics | | |
| Number of server roundtrips | 1 → | 1.0000 |
| TDS packets sent from client | 1 → | 1.0000 |
| TDS packets received from server | 1 → | 1.0000 |
| Bytes sent from client | 100 → | 100.0000 |
| Bytes received from server | 256 → | 256.0000 |
| Time Statistics | | |
| Client processing time | 1 → | 1.0000 |
| Total execution time | 34 → | 34.0000 |
| Wait time on server replies | 33 → | 33.0000 |

From the bottom section of this tab, we can see where the time was spent. In this case, out of a total of 34 milliseconds for the query, 33 milliseconds was spent waiting for the server.

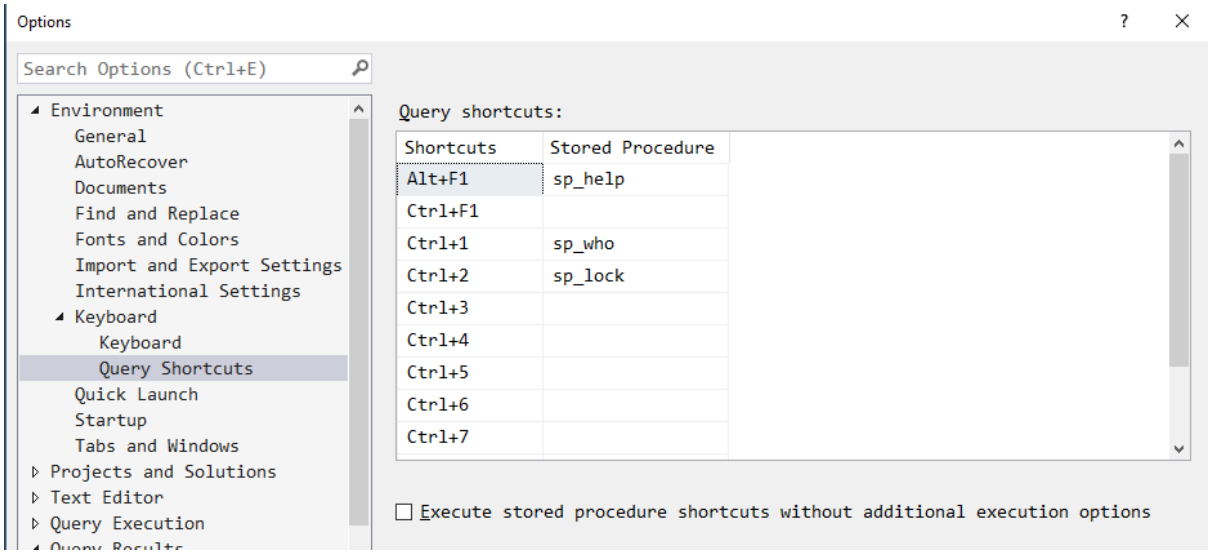
## 4.8 Set shortcuts for favorite stored procedures

In an earlier post, I mentioned how useful the F1 key is. On its own, it provides syntax help, but when you highlight an object and hit Alt-F1, you get to see metadata about the object.

Under the covers, this just runs the sp_help stored procedure. Alt-F1 has been mapped to that.

You can see where this is configured, change it if required, and/or configure other procedures as well.

In **Tools**, **Options**, **Environment**, **Keyboard**, then **Query Shortcuts**, you can see this:




Here we can see how the Alt-F1 command is configured. By default, only three shortcuts are configured. It's easy to configure another one for your own use.

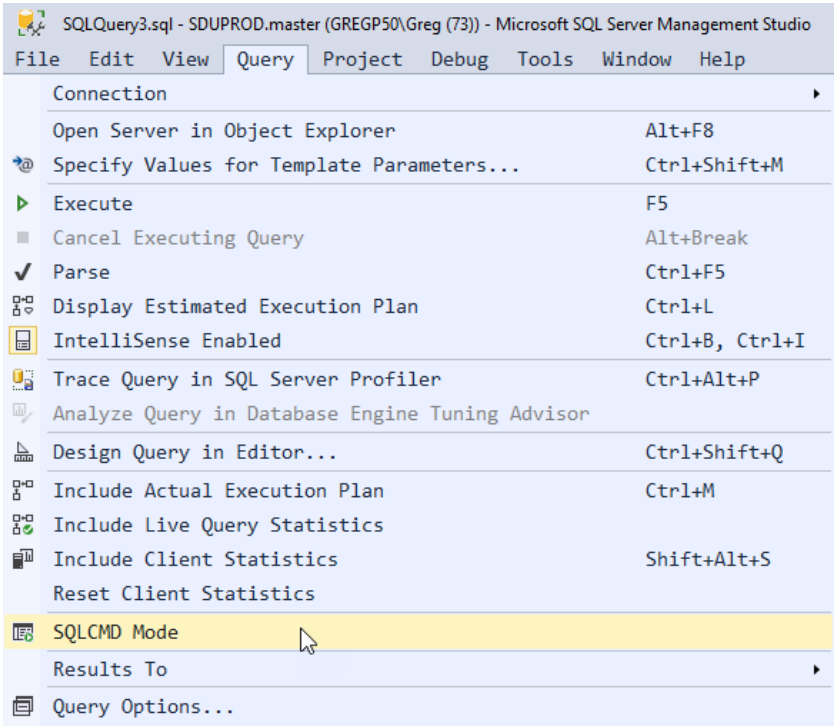
## 4.9 Set SQLCMD mode for all new windows

SQLCMD mode changes how queries are executed in SQL Server Management Studio (SSMS). When using this mode, you can work with options that aren't normally part of SQL Server T-SQL scripts.

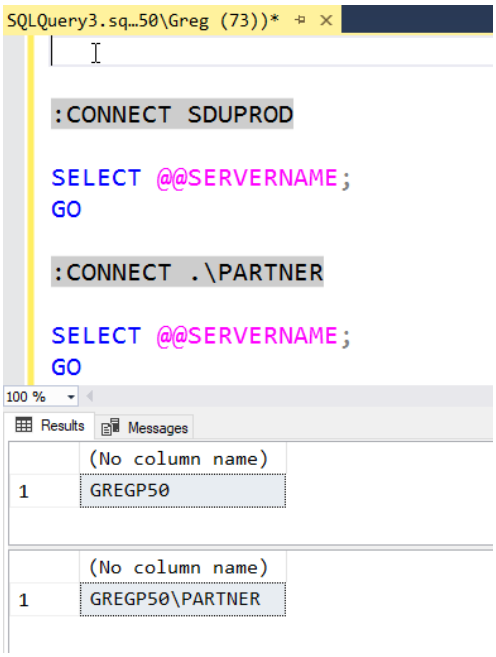
Some installation scripts also require SQLCMD mode and will fail if it's not enabled.

Here's an example of running a query against 3 servers within the same script:

First we open a new query window, then on the Query menu, we choose SQLCMD Mode:



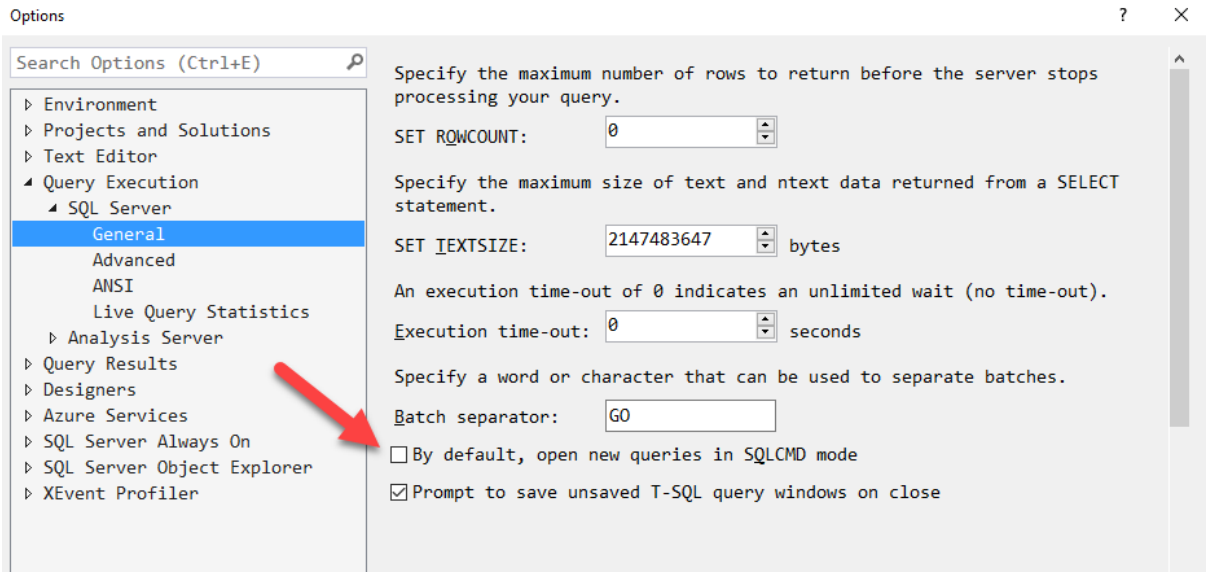

Then I execute the following query:



Note the :CONNECT command is used to connect to another server.

Because everything else works pretty much the same, and you get a whole lot of additional options, you might choose to open all your new queries in SQLCMD mode. That's easy to do.

In Tools, Options, Query Execution, SQL Server, then General, there is a checkbox to enable this.

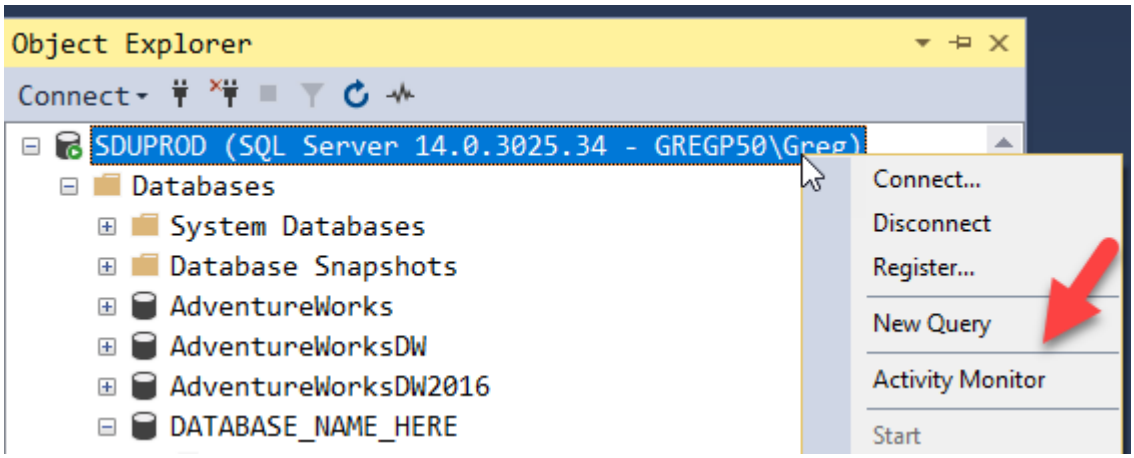


### 4.10 Activity Monitor

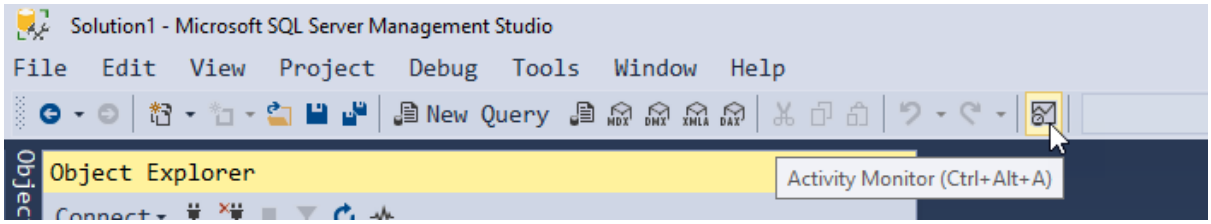
This is a quick tip but an important one. I see many people using SQL Server Management Studio (SSMS) and they aren't aware of Activity Monitor.

While there are many clever things that we can do with queries, to interrogate the health of the system, don't forget that there is quite a bit of useful information in Activity Monitor, and it's easy to get to.

There are two basic ways to launch Activity Monitor. The first is to right-click the server in Object Explorer:



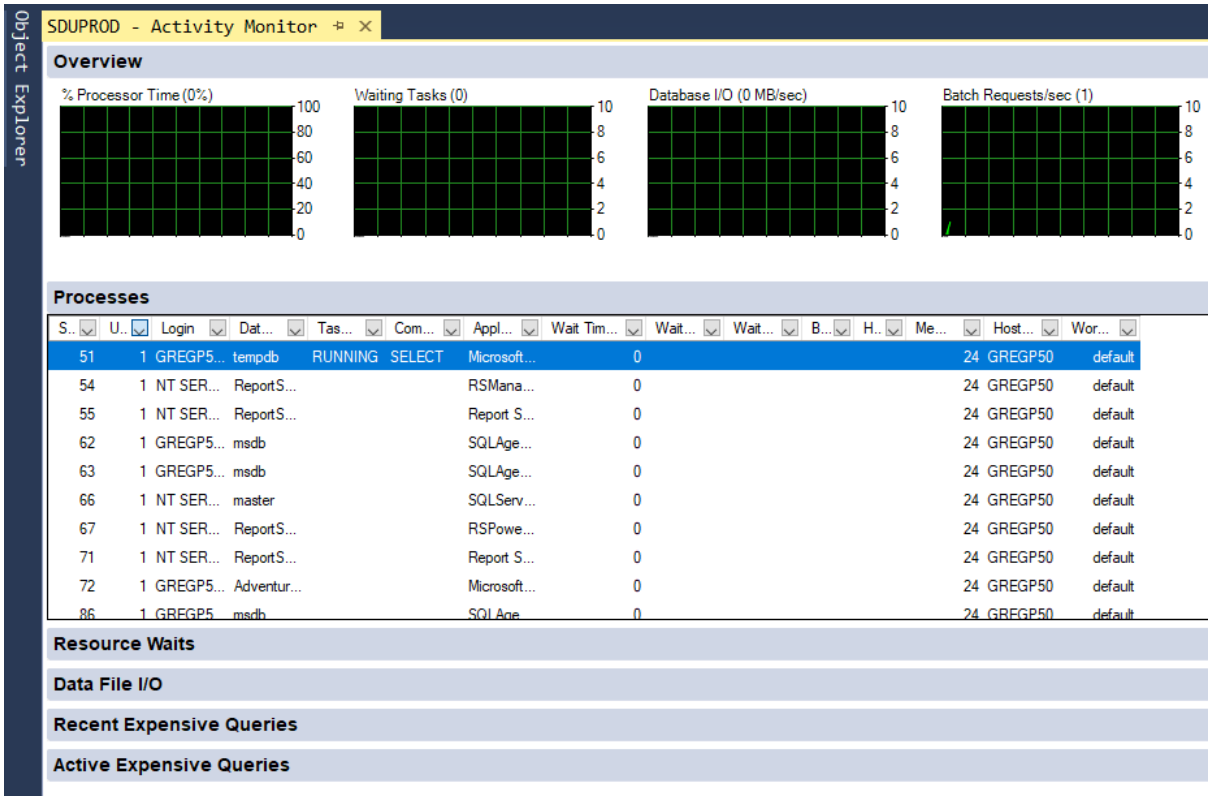

The other common way to launch it is from the Toolbar:




Note that if you connect to more than one server in Object Explorer, Activity Monitor will connect to whichever one you have selected any object from within.

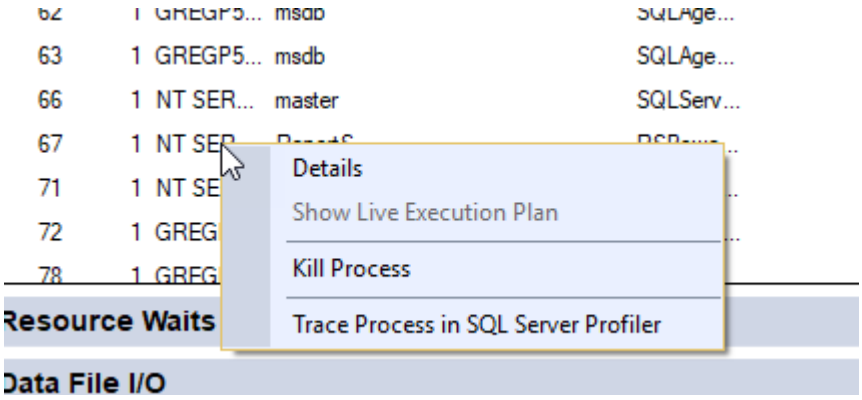
**Activity Monitor puts a bit of a load on the server that it's connected to but I generally don't find it too bad. However, please don't leave it running and go on using other tabs. I've been to sites where there are many copies of it running all the time from several users. Don't do that.**

I don't find most of the graphs at the top very useful, apart from perhaps the processor time.



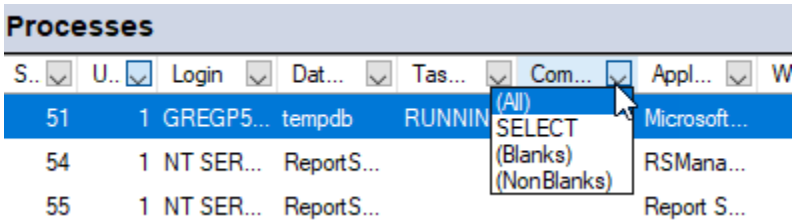

It will show you if the server is running flat out.

The list of Processes is more interesting. If you right-click any session, you get these options:




The Details link will show you the last command executed on that connection. Take note that this doesn't mean it's still running. You can also kill the process (obviously carefully), and you can connect SQL Server Profiler to the server and filter the session immediately, to see what it's doing.

The columns are filterable.




They show you a list of values currently in that column, plus an All, and a choice of Blanks (rows with no value in this column) or NonBlanks (rows with anything in this column). They start as All.

For a simple example of using this though, we could pick sessions that have any type of command running, by choosing Task State of RUNNING.

**Processes**

| S.. | U.. | Login | Dat... | Task State | Com... | Appl... | Wait Tim... | Wait... | Wait... | B... | H.. | Me... | Host... | Wor... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 51 | 1 | GREGP5... | tempdb | RUNNING | SELECT | Microsoft... | 0 | | | | | 24 | GREGP50 | default |

One that I often use this view for is to look for blocking issues. Every process that's blocked by another process will tell you that. Generally, what I'm looking for is the head of a blocking chain ie: who's the main culprit that's blocking everyone.

For that, I look for a value of 1 in the Head Blocker column. Unfortunately, the way it's designed, you can't select that value until there is a row with that value.

The Application Name, Database Name, and Login can all be pretty useful as well.

The Resource Waits section is only mildly interesting.

**Processes**

**Resource Waits**

| Wait Category | Wait Time (ms/sec) | Recent Wait Time (ms/sec) | Average Waiter Count | Cumulative Wait Time (sec) |
|---|---|---|---|---|
| Buffer I/O | 0 | 0 | 0.0 | 3 |
| Buffer Latch | 0 | 0 | 0.0 | 0 |
| Latch | 0 | 0 | 0.0 | 0 |
| Lock | 0 | 0 | 0.0 | 24 |
| Logging | 0 | 0 | 0.0 | 24 |
| Memory | 0 | 0 | 0.0 | 0 |
| Network I/O | 0 | 0 | 0.0 | 0 |
| Other | 0 | 0 | 0.0 | 0 |

The information there is at a bit of a coarse level to be really useful to me. Note that on this system, Buffer I/O is top of the list, but the cumulative wait time (since the server restarted) is small. Over time, if the system has been up for a long time, you can start to get a feel for the main waits in here, but be aware that there are a lot of values that can appear in here, without actually being an issue.

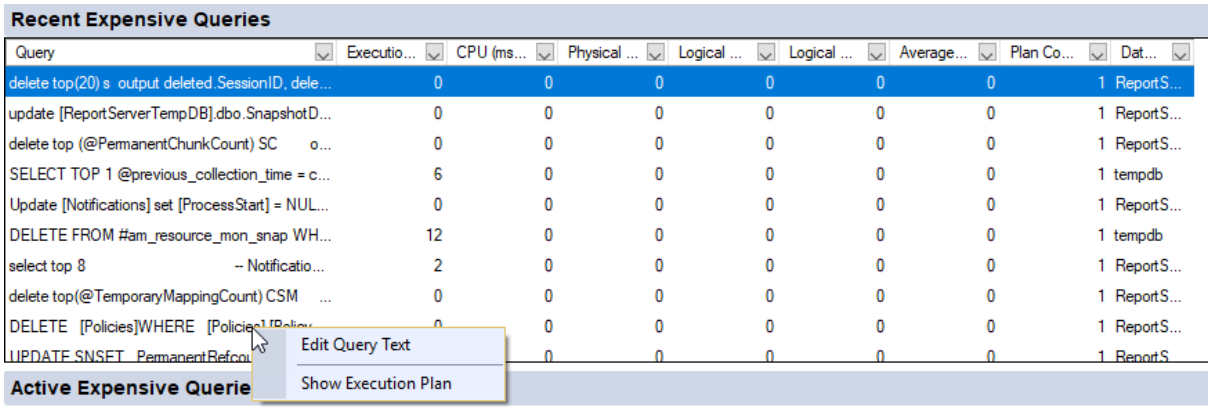
The Data File I/O list is a little more interesting:

**Data File I/O**

| Database | File Name | MB/sec Read | MB/sec Written | Response Time (ms) |
|---|---|---|---|---|
| tempdb | C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MS... | 0.0 | 0.0 | 8 |
| AdventureWorks | C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MS... | 0.0 | 0.0 | 0 |
| AdventureWorks | C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MS... | 0.0 | 0.0 | 0 |
| AdventureWorks... | C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MS... | 0.0 | 0.0 | 0 |
| AdventureWorks... | C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MS... | 0.0 | 0.0 | 0 |
| AdventureWorks... | C:\Temp\AdventureWorksDW2016_Data.mdf | 0.0 | 0.0 | 0 |
| AdventureWorks... | C:\Temp\AdventureWorksDW2016_Log.ldf | 0.0 | 0.0 | 0 |
| DATABASE_NA... | C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MS... | 0.0 | 0.0 | 0 |
| DATABASE_NA... | C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MS... | 0.0 | 0.0 | 0 |

This will show you how busy each data and log file is, for all databases. I generally sort it by Response Time (ms) descending. The value here is then basically the latency for the I/O on that file. In this example, it's 8 milliseconds. That's ok.

The Recent Expensive Queries list is interesting. The information is available from the system DMVs but this puts some useful data in an easy to get location:



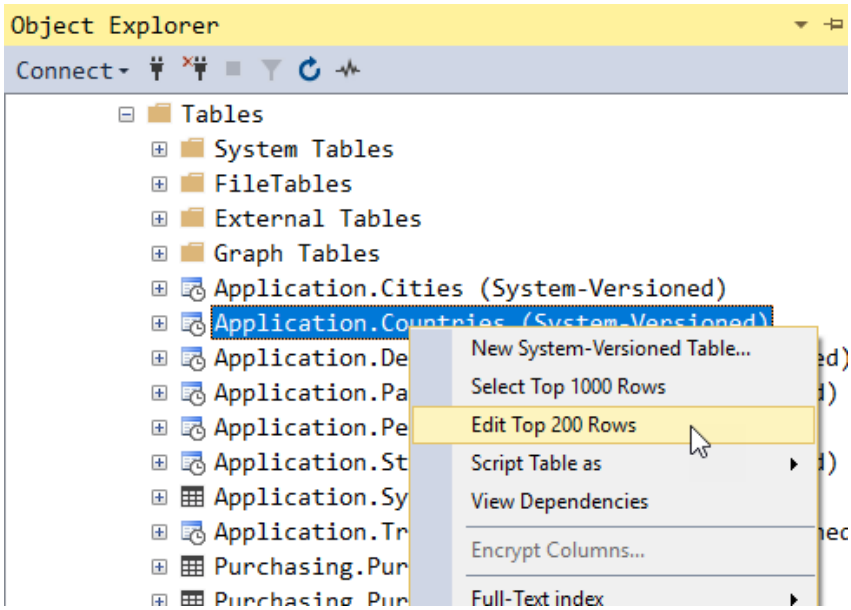

It keeps updating this over time. Note that this won't be showing you queries currently running, just ones that were expensive and finished recently. If you right-click one, you can either look at the query text, or check out the execution plan that was being used.

The final section with Active Expensive Queries will only have data if you're using Live Query Statistics. I'll write about it another day.

# 5 Results

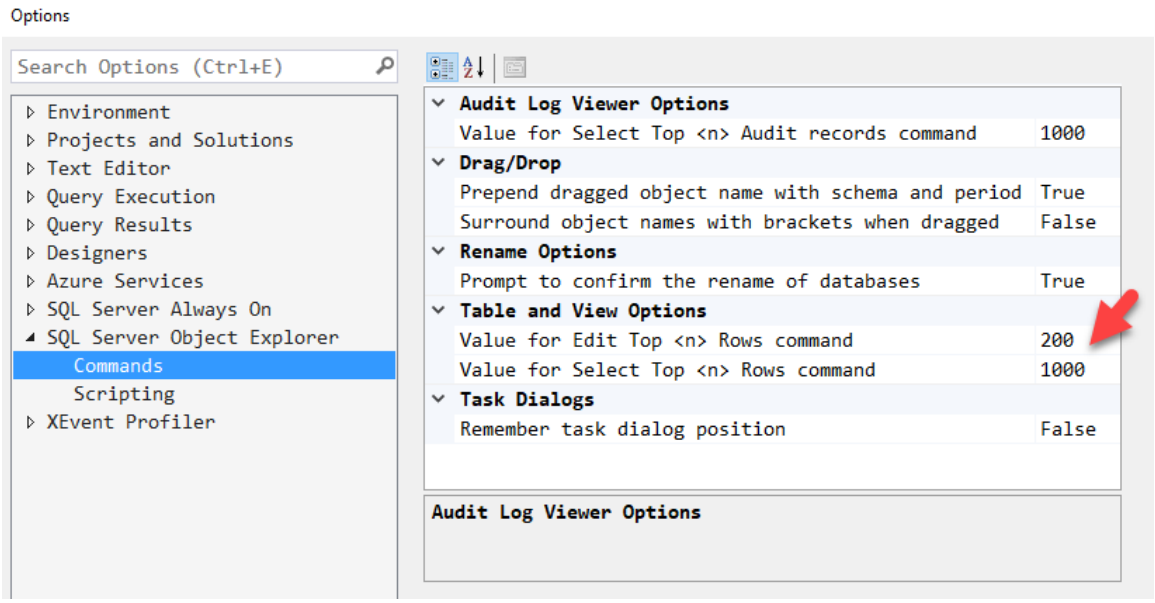
## 5.1 Change the number of rows selected or edited in Object Explorer

When you right-click a table in SQL Server Management Studio, you get options for selecting or editing but the number of rows is limited:




Those values can be changed. By default, these numbers are both 200, but I've decided to change the default number of rows selected to 1000.

In Tools, Options, SQL Server Object Explorer, then Commands, you can set the values to whatever suits you:




I don't tend to ever use the Edit option but I'd suggest not making it too large.

## 5.2 Viewing and configuring spatial data output

SQL Server 2008 added the ability to work with spatial data by the additional of the geometry and geography data types. When they first were added, there was no tools support for working with them, and all we had was direct manipulation of their internal binary storage.

Here's an example:

```
-- Draw a square
DECLARE @Shape GEOMETRY;
SET @Shape = GEOMETRY::STGeomFromText('POLYGON ((10 10, 10 40, 40 40, 40 10, 10 10))',0);
SELECT @Shape AS Square;
GO
```

100 %

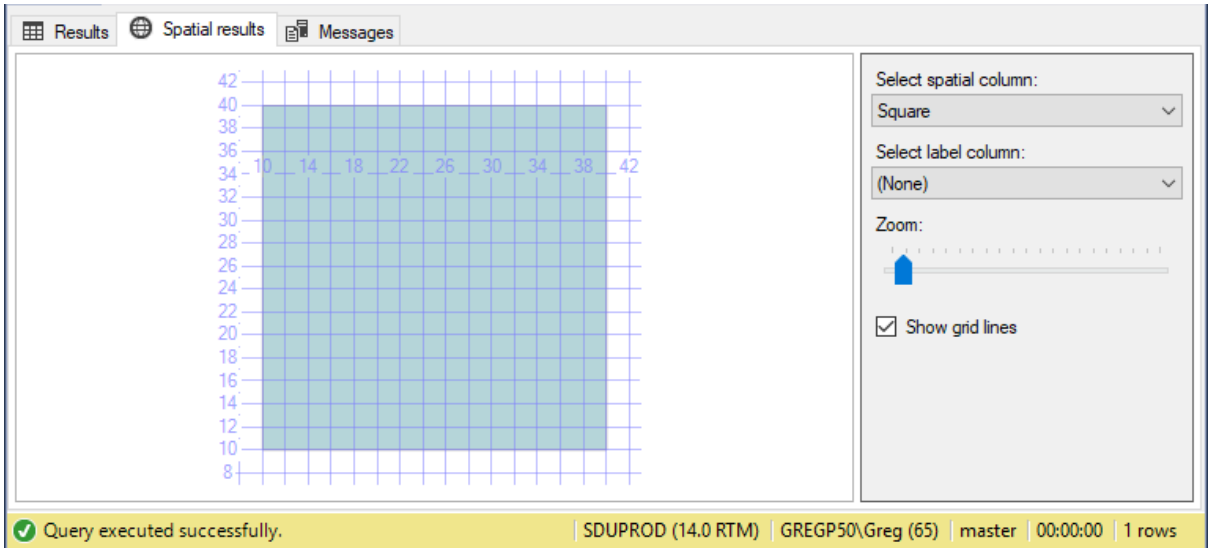
Results | Spatial results | Messages

| | Square |
|---|---|
| 1 | 0x0000000001040500000000000000000002440000000000... |

I've defined a variable named @Shape of type GEOMETRY. I've then assigned a shape to it, based on a polygon formed by a set of points. If you look carefully, you'll notice that it's a square.

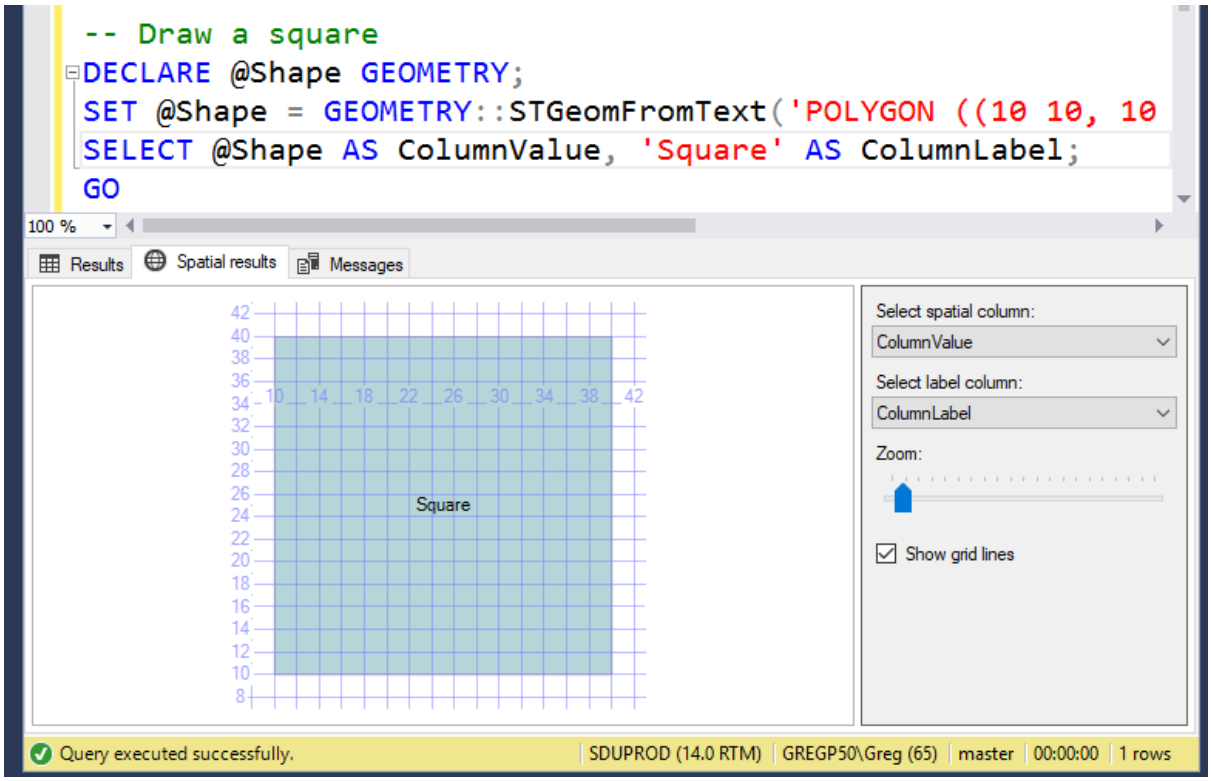
But when I select the value, what is returned is the internal value of the data type, as a binary string, **written in hexadecimal for ease of reading** 😊

And at first, that's all we had. But notice that there is now another results tab:




As soon as SSMS sees any spatial data in the results, it adds a tab to try to visualize it. Under the covers, it's actually the mapping control that Microsoft purchased from Dundas and put into Reporting Services.

On the right hand side, we can pick the column to be displayed, because we might have more than one, and we can overlay a label if another column is holding the label. In this case, let's modify the query to have two columns, one with the value, one with the label:




The mapping tool works out where to put the label so it's somewhere on the image. SQL Server has a function call to try to help with that. (It's easy for a square, but hard for say a donut).

If more than one row of data is returned, a shape is displayed in a different color for each value. This can lead to some awesome outcomes. Note what happens if we select all of the Application.Countries table from WideWorldImporters, choose the CountryName as the label, and zoom in slightly:

This is an awesome tool for visualizing data.

### 5.3 XML editor and Increase XML output size

Most people use SQL Server Management Studio (SSMS) to edit SQL queries. No big surprise there. The files will have a file type of **.sql**.

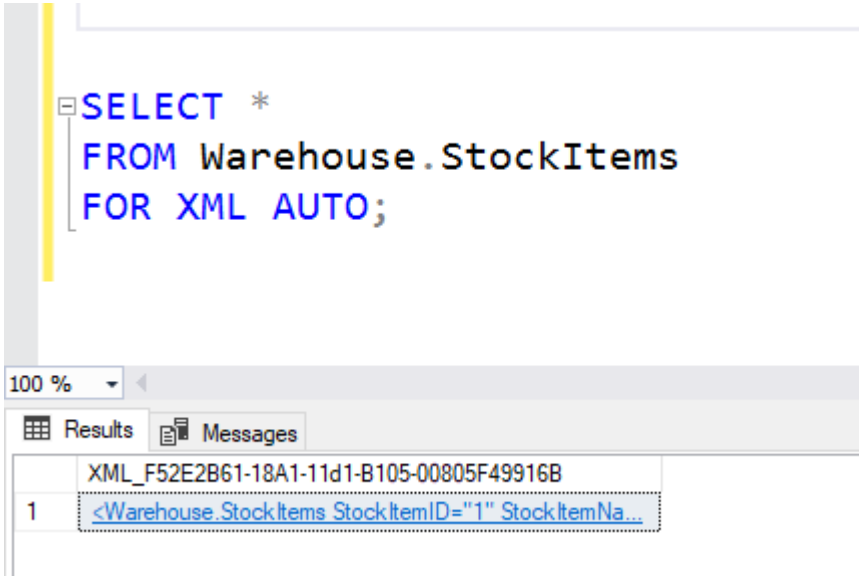
But what many people don't understand is that SSMS inherits many of its underlying Visual Studio's abilities to edit other document types.

For example, if you open a **.txt** text file, you can edit it just fine, and you can also include files like this in SSMS script projects. That can be useful for additional notes and documentation.

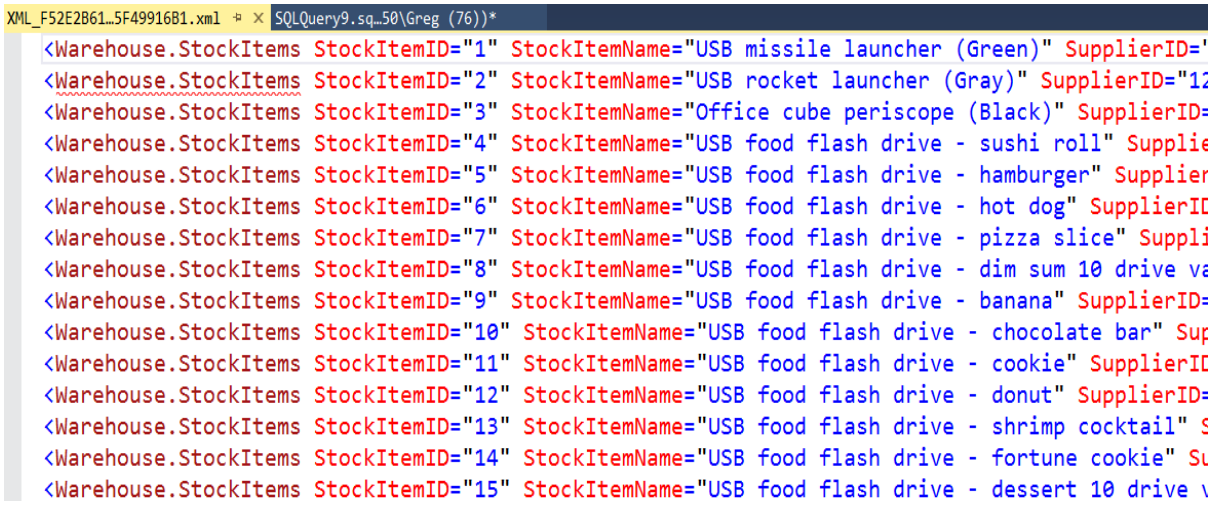
SSMS also knows how to open related file types like **.sqlplan** (for query plans) and **.xdl** files (for deadlocks), and more.

Most of these other file types though, are actually XML files with specific schemas constraining their contents. **SSMS also contains a perfectly acceptable XML editor.**

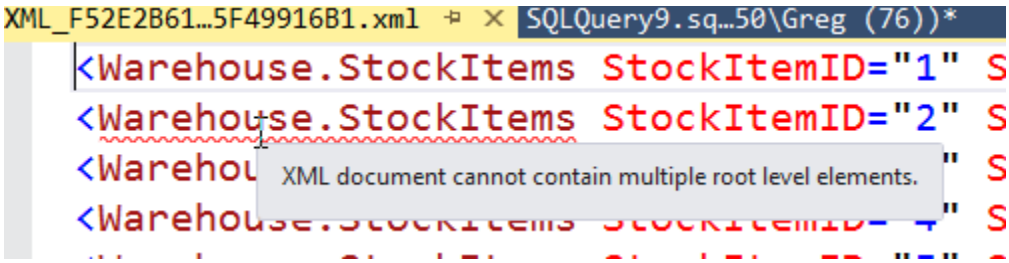
Here's an example:




If I execute the above query, the outcome is some XML. Note that SSMS recognizes that the output data type is XML and then provides a hyperlink for opening it. If I click on the link, I see this:

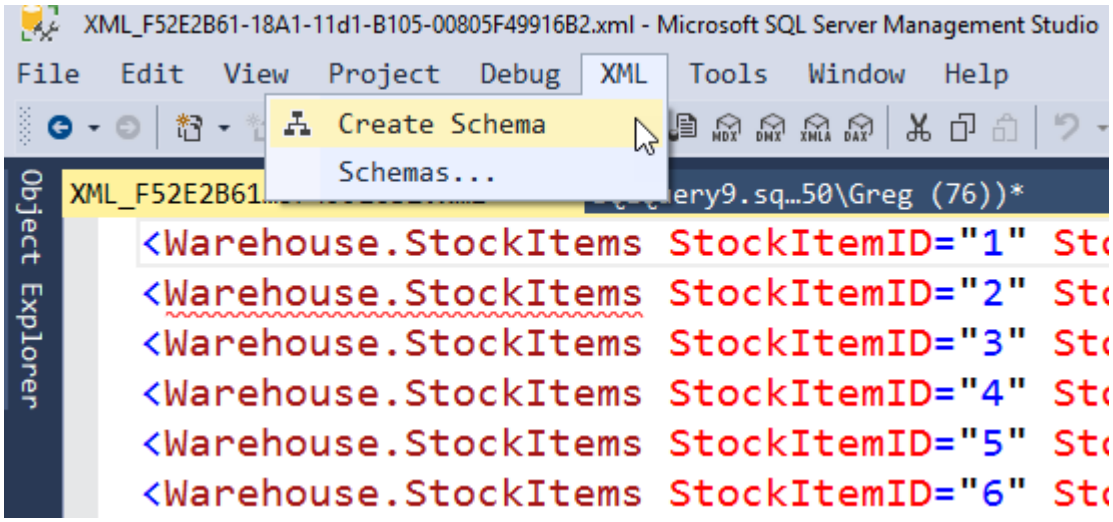


The important item to notice here though is the red squiggly on the second line. If we hover over that, we'll see this:

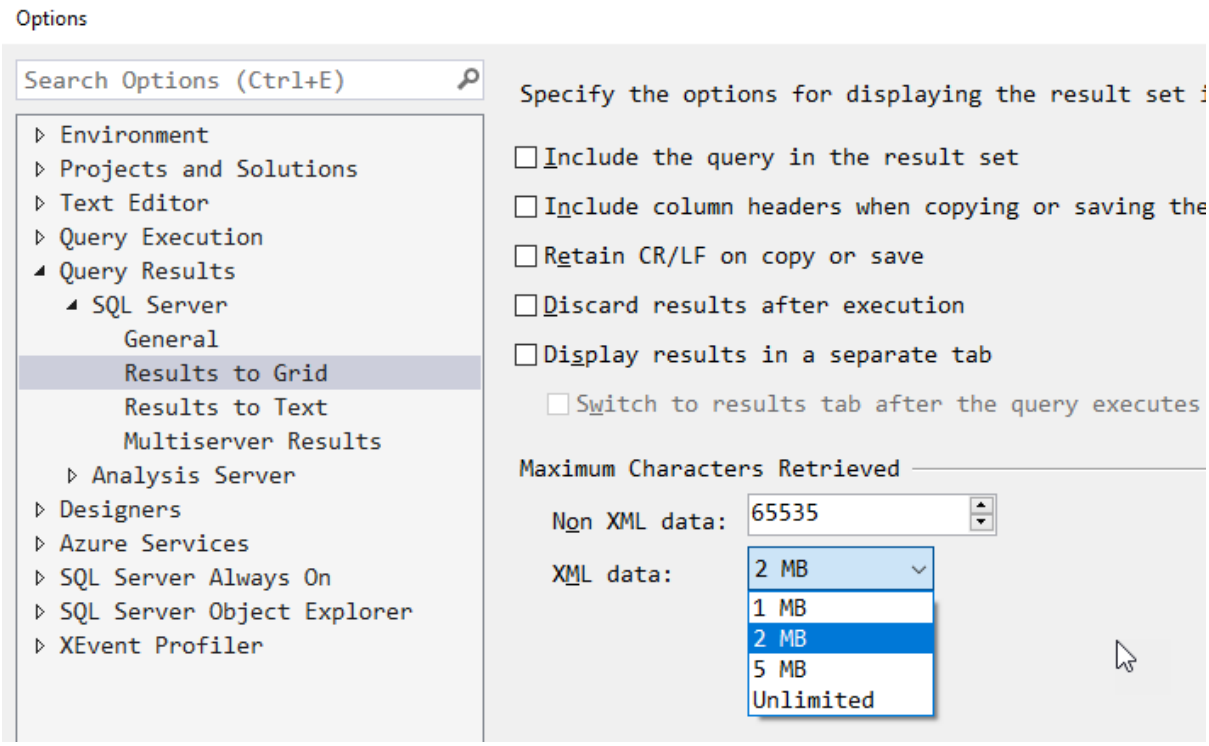



XML documents can only have a single root element. This XML is actually a fragment, not a complete document, and so it thinks that all the stock item lines are all root elements.

The important thing is that this is an XML editor, not just an XML viewer. Notice that when an XML file is open, an XML menu also appears:




Now, while it's not a bad XML editor, it has limits on the size of the data that you can work with, but you can control that too. In Tools, Options, Query Results, SQL Server, Results to Grid, you can see this:

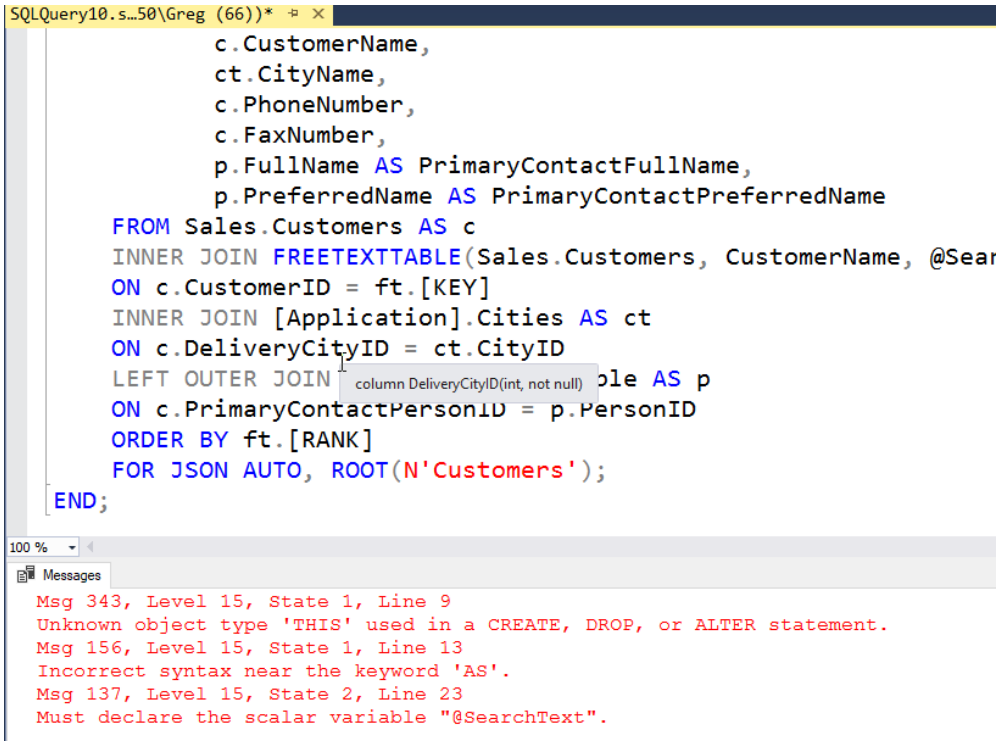



By default, you are limited to 2MB of XML data. You can increase this to unlimited but keep in mind that SSMS is (unfortunately) still a 32 bit application and can struggle to work with gigantic files.

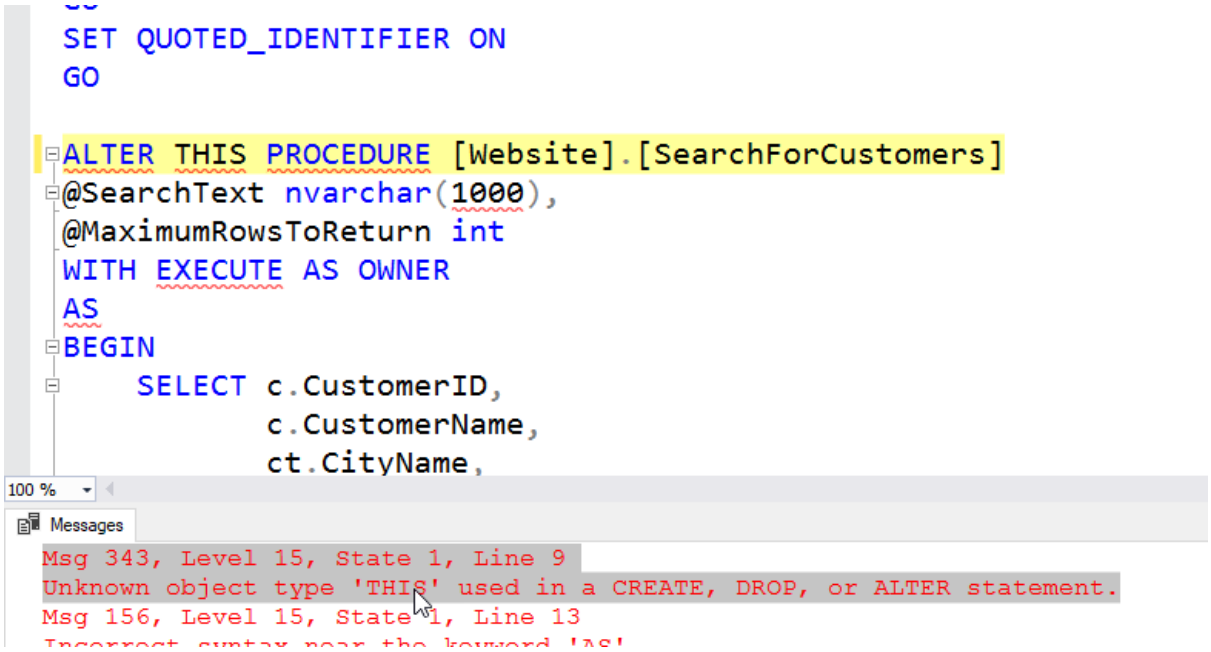
## 5.4 Find error locations within scripts

This is probably one of the simplest tips that I've published, yet I'm endlessly surprised how many people do not realize that it's available.

When you have a script loaded in SQL Server Management Studio (SSMS), and you execute the script, you might run into an error like this:




To find where the error is, just double-click the error down in the Messages tab. I double-clicked it, and it took me directly to the error and highlighted it.



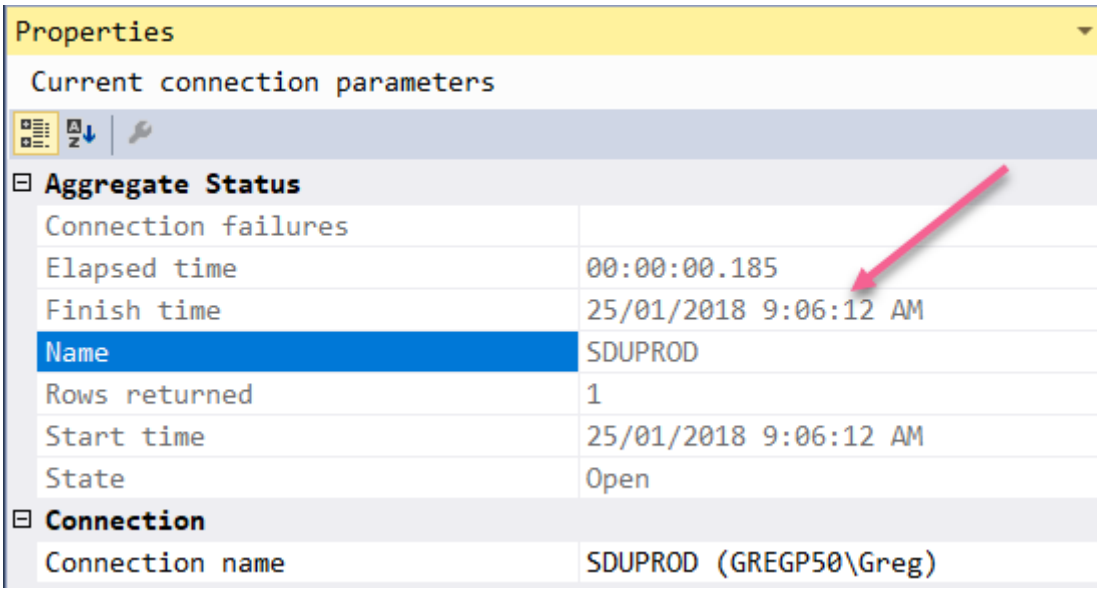
## 5.5 When did my query finish?

It's likely that everyone who uses SQL Server Management Studio (SSMS) knows how to tell how long a query ran for. You can see it in the bottom right of the status bar when a query finishes.

But one question that often comes up with a long-running query is **when did my query finish?**

That's not in the status bar and many people don't seem to be aware that you can find it out.

It's part of the data in the Properties window. So when you come to a query window where the query has finished, and you're wondering when it finished, hit F4 (or right-click in the window and click Properties), and you'll see this info shown:



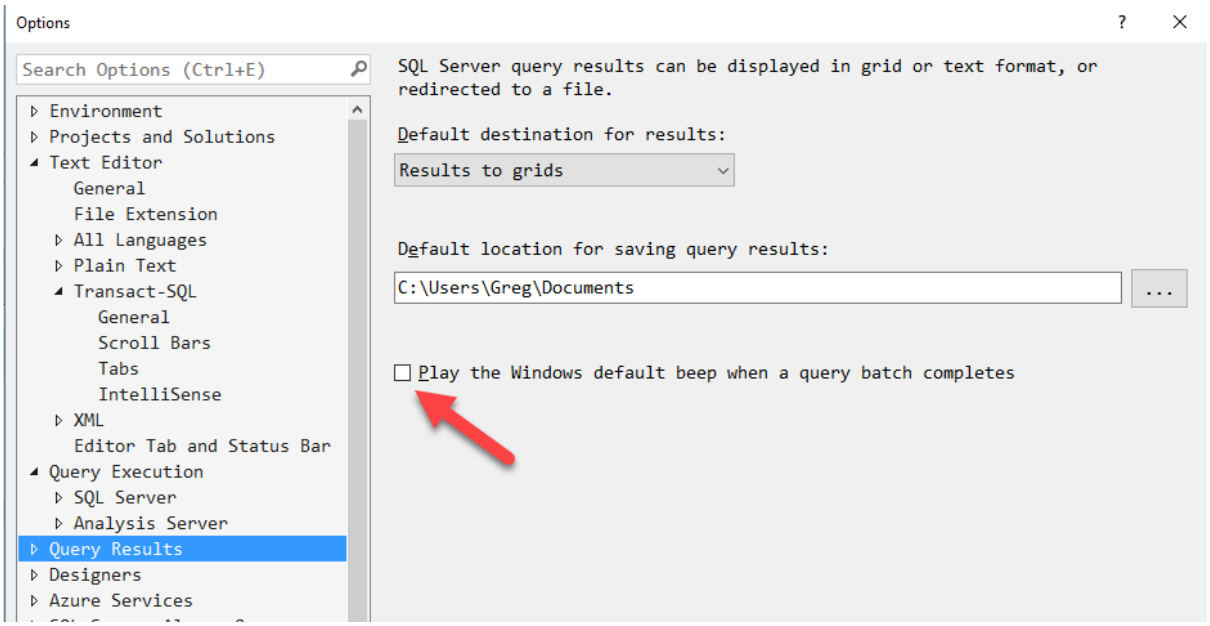
## 5.6 Play a sound when a query completes

In another section, I mentioned that when a long running query completes, I might not have been waiting around for it, and so I wanted to know when it completed.

But sometimes I do wait around for a query to complete, yet I'm distracted by other things and don't realize that the query has actually completed. That's not surprising because if a query takes a long time, I'm probably going to go on with other work while that's running.

So I want to get a prompt when the query finishes.

SQL Server Management Studio (SSMS) does provide an option for this. In Tools, Options, Query Results, there is an option to **Play the Windows default beep when a query batch completes**.




I do wish it was a stronger option than this but at least it's a start.

What I'd particularly like would be:

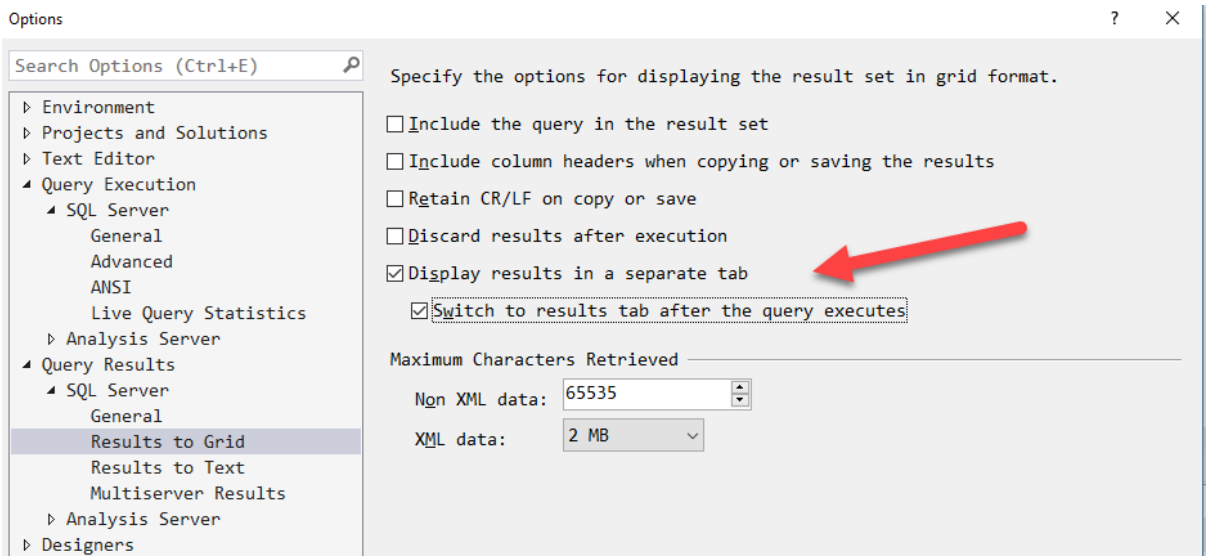
- Ability to play a different sound, not just the default beep.
- Ability to enable/disable this on a specific query window once a query is already running.

Having this on all the time would be quite annoying, so I'd be pretty selective about using it in its current form.

## 5.7 Query and results in separate tab

Another simple tip for today. In SQL Server Management Studio (SSMS), query results are normally shown at the bottom of the query window.

This can greatly reduce the screen real estate both for the query, and for viewing the results.




In **Tools**, **Options**, **Query Results**, **SQL Server**, **Results to Grid**, there is an option to **Display results in a separate tab**. This can be very useful and generally you will also want to choose the extra option to **Switch to results tab after the query executes**.

There is a similar (but separate) option for working with results to text rather than grid.

One further option on this dialog that's worth mentioning is the **Include the query in the result set**. I find that useful when working with text output, not so much with grid output.

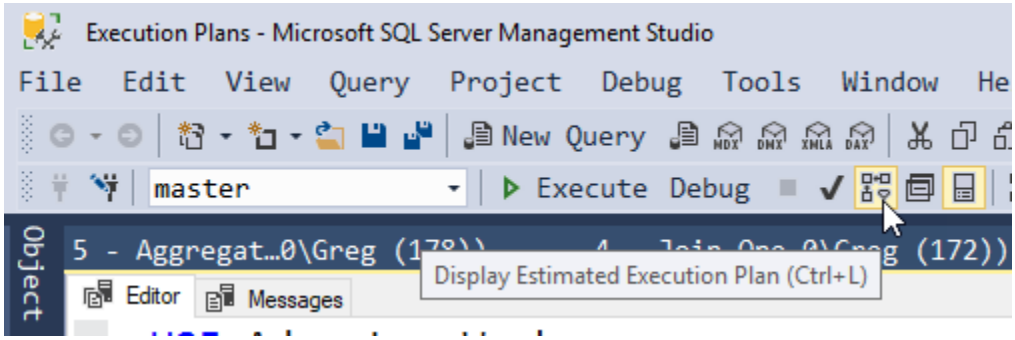
# 6 Execution Plans

## 6.1 Compare query plans

One of the advantages of SQL Server Management Studio (SSMS) is that it can be used to analyze queries, not just to execute them.

There are two basic types of query plan: estimated execution plans, and actual execution plans.

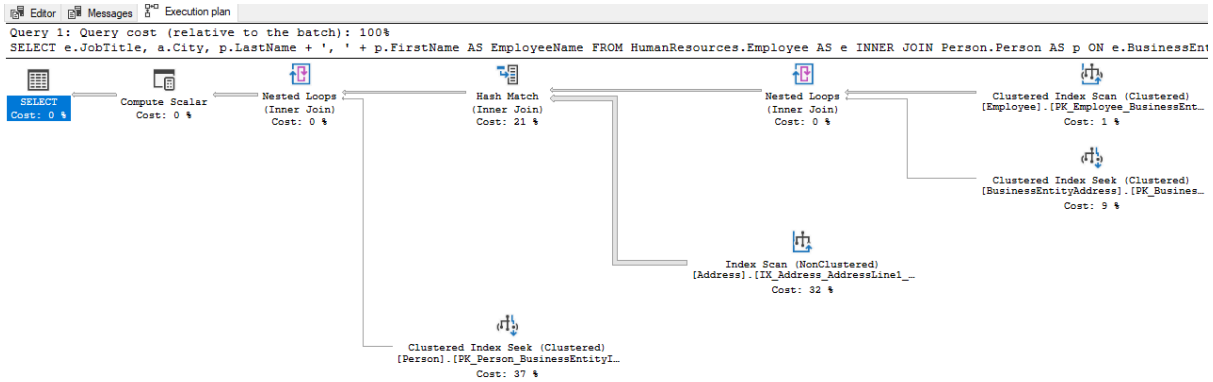
For a typical query, I can obtain the estimated execution plan, by hitting Ctrl-L, choosing the option in the Query menu, or clicking on the toolbar icon:




Let's do this for the following query:

```
-- Hash Match

SELECT e.JobTitle, a.City, p.LastName + ', ' + p.FirstName AS EmployeeName
FROM HumanResources.Employee AS e
INNER JOIN Person.Person AS p
ON e.BusinessEntityID = p.BusinessEntityID
INNER JOIN Person.BusinessEntityAddress AS bea
ON e.BusinessEntityID = bea.BusinessEntityID
INNER JOIN Person.Address AS a
ON bea.AddressID = a.AddressID;
```

We get the following plan output:



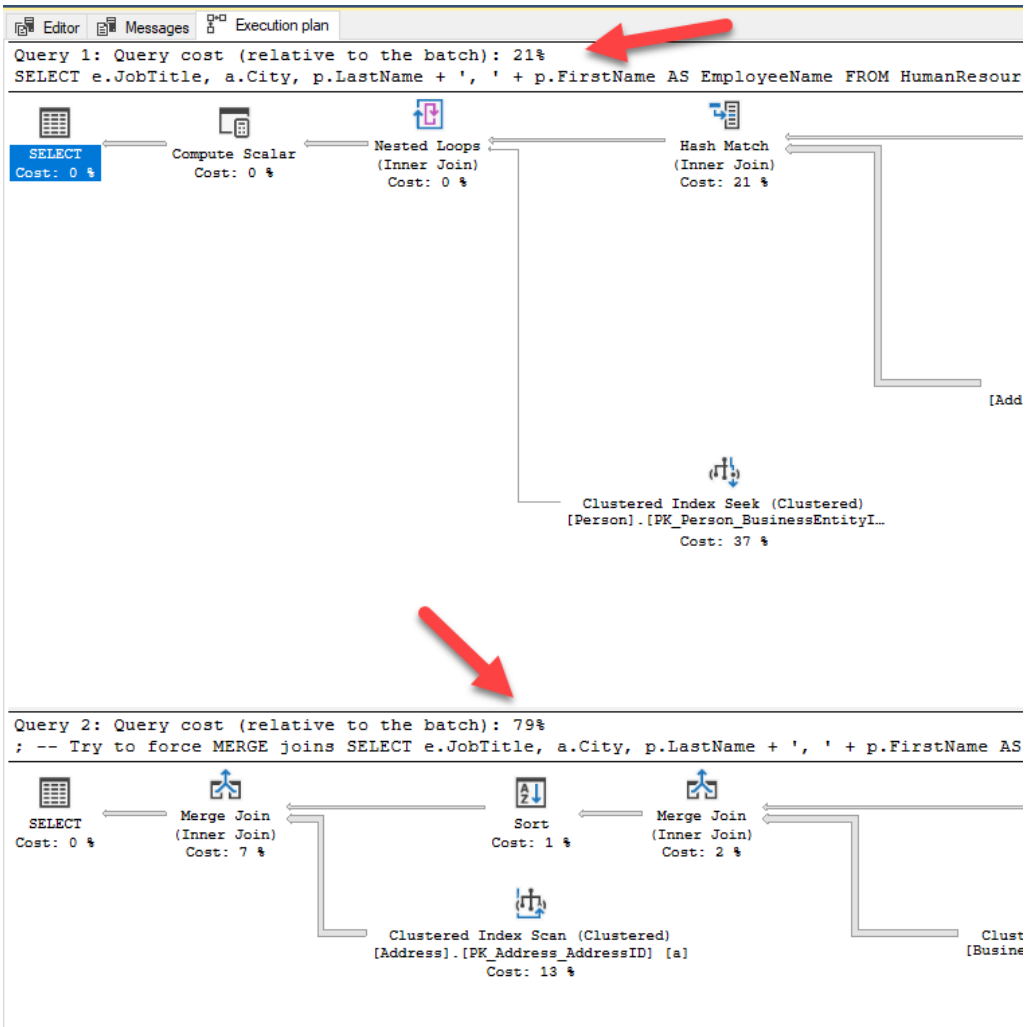

Now we might look at this plan and wonder if we exerted control over how the joins were performed, if we'd improve things. Perhaps we'd heard that merge joins were more efficient and thought SQL Server should have used those.

Now we can change the query like this:

```
-- Try to force MERGE joins

SELECT e.JobTitle, a.City, p.LastName + ', ' + p.FirstName AS EmployeeName
FROM HumanResources.Employee AS e
INNER MERGE JOIN Person.Person AS p
ON e.BusinessEntityID = p.BusinessEntityID
INNER MERGE JOIN Person.BusinessEntityAddress AS bea
ON e.BusinessEntityID = bea.BusinessEntityID
INNER MERGE JOIN Person.Address AS a
ON bea.AddressID = a.AddressID;
```

Notice I've added the word MERGE between INNER and JOIN in this query. But how do we know what SQL Server thinks? We can get another estimated query plan but what we really want is to compare them.

When we obtain an estimated plan for multiple queries at once, SSMS shows us a comparison of the two, by showing us the proportion of the overall query for each part. (Perhaps we shouldn't force that change 😊)



## 6.2 Missing index details

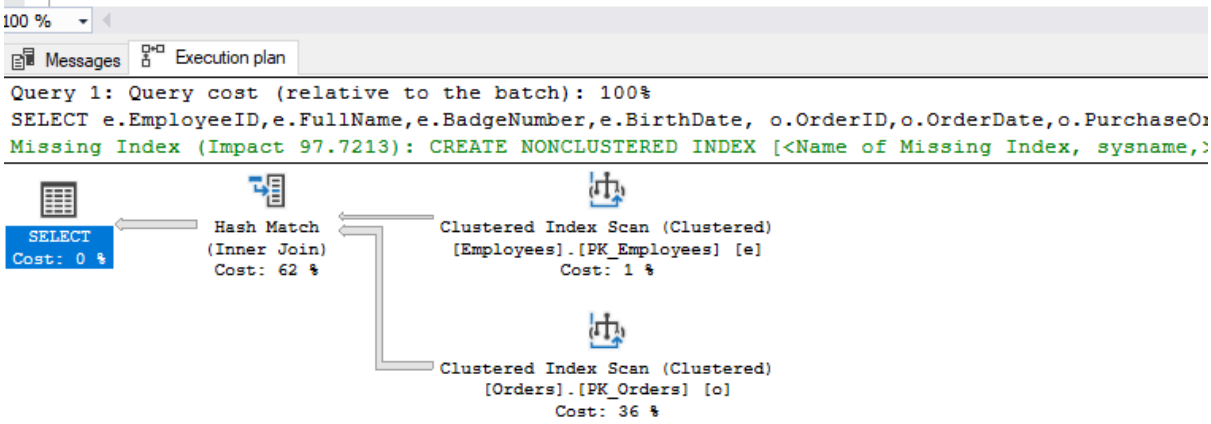
I've mentioned before that SQL Server Management Studio (SSMS) is a good tool for analyzing queries, as much as for executing them.

In SQL Server 2005, query plans had missing index details added. When a query plan was created, SQL Server recorded that it thought it could have executed the query better, if only you'd provided it with appropriate indexes. But at that point, the suggestions weren't very good, and the tools didn't show them.

In SQL Server 2008, the suggestions got better (eg: we weren't endlessly having suggestions to create non-clustered indexes for every column in the table), and SSMS now showed them clearly.

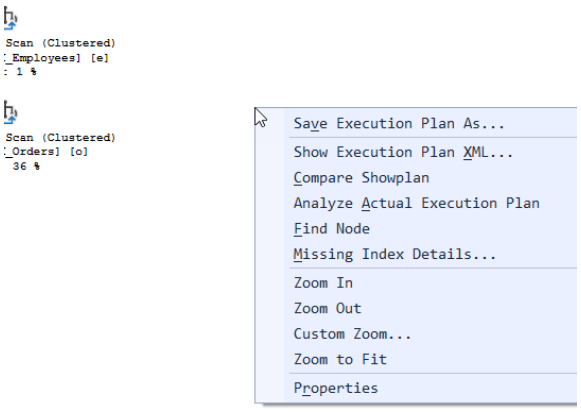
Here's an example query with an issue:

```
-- Is there an index missing for this query?

SELECT e.EmployeeID,e.FullName,e.BadgeNumber,e.BirthDate,
       o.OrderID,o.OrderDate,o.PurchaseOrderNumber
FROM dbo.Employees AS e
INNER JOIN dbo.Orders AS o
ON e.EmployeeID = o.EmployeeID;
```

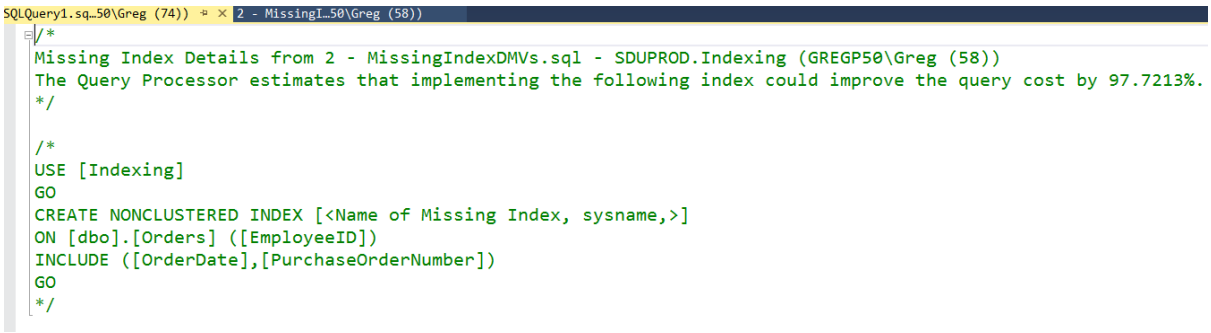



SQL Server thinks it's doing this work the hard way. Note that it suggests an impact of over 97% on running this query, just because an index is missing.

To find out what it wants, we right-click in the white area of the query plan:



The option of interest is **Missing Index Details**. When we click that to open it, the following appears:

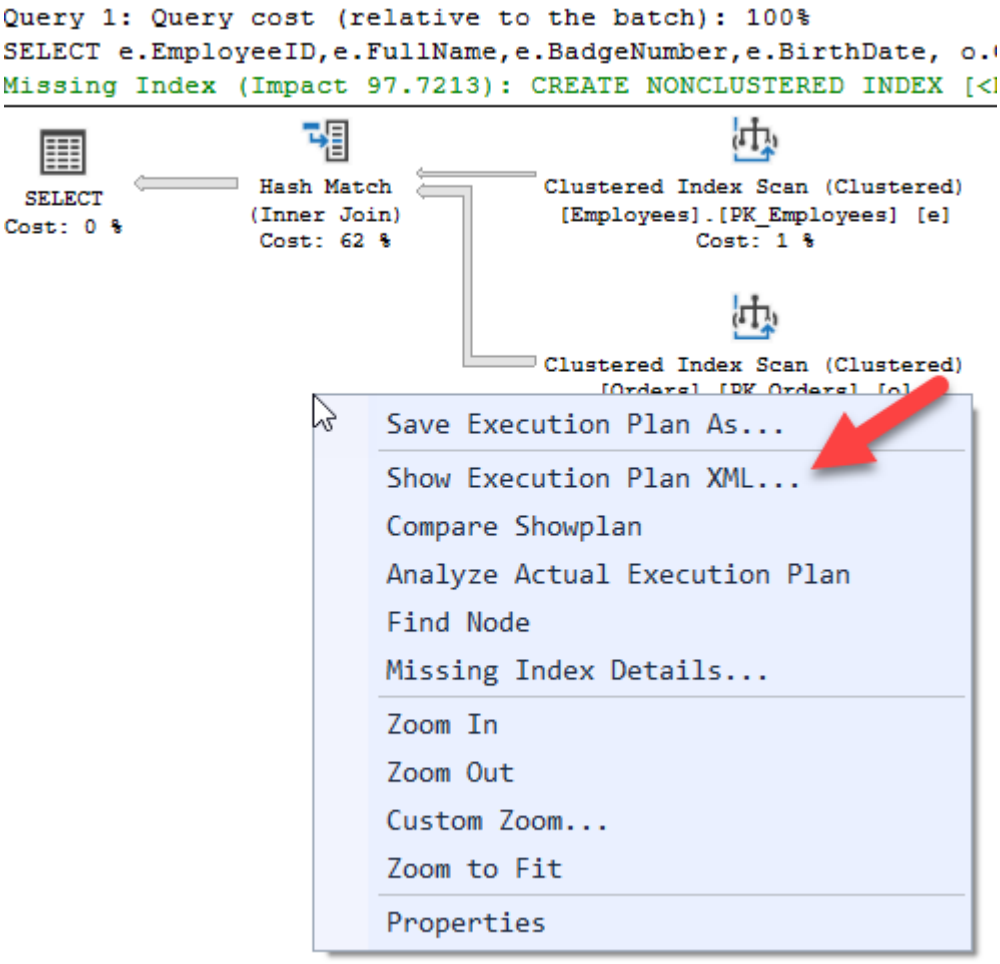



Overall, the suggestions that this system makes aren't bad. They are far from perfect but if you don't know much about SQL Server indexing, these suggestions might make a good start. Just don't blindly follow large numbers of them. Someone with strong SQL Server skills can often come up with better suggestions than those currently offered by the tools.

Also, make sure you rename the index. I have come across indexes at customer sites where the index name is **[<Name of Missing Index, sysname,>]**. I wish I was joking.

## 6.3 Saving and sharing query plans

Currently, SQL Server query plans are stored as XML. You can see what they look like by right-clicking in any query plan in SQL Server Management Studio (SSMS), and clicking Show Execution Plan XML:




That will return a whole bunch of XML like this:


It's important to understand that when SSMS is showing a graphical execution plan, it's just graphically rendering some XML like the plan above.

The Properties window in SSMS is also showing details extracted from that same XML.

Prior to SQL Server 2005, it was very difficult to share a graphical query plan with anyone else. You'd have to pop up the details of each operator one by one and take screenshots. Nasty.

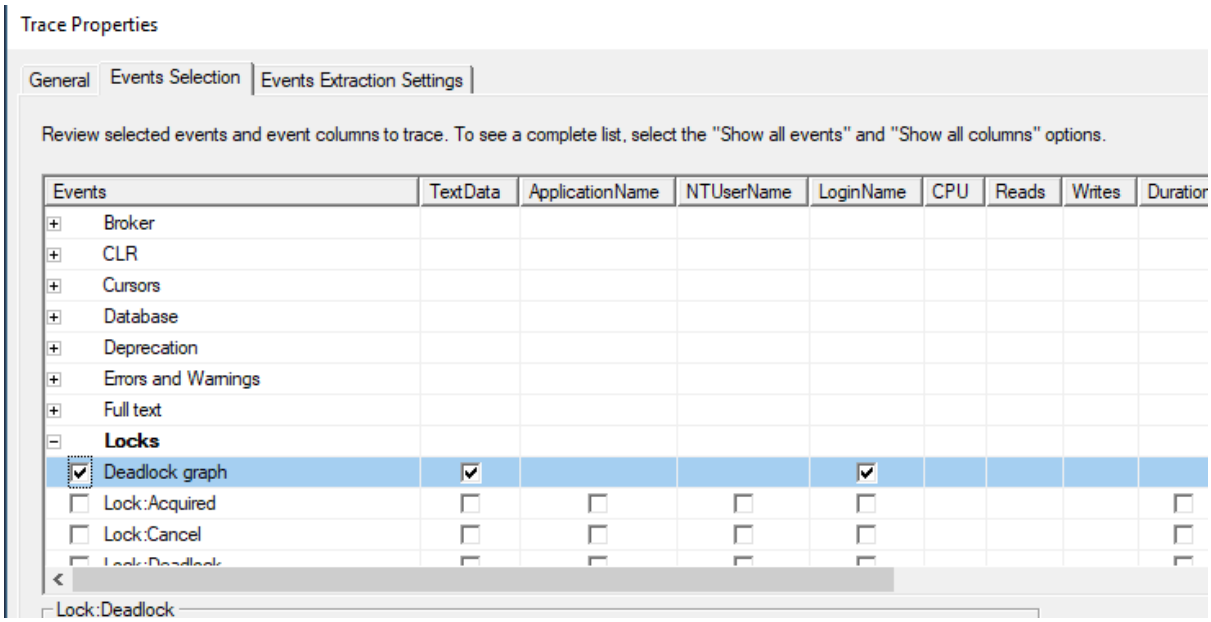
Since SQL Server 2005, query plans can be stored as XML files, with a **.sqlplan** file extension. If you save a plan, that's the default file extension. And because SSMS is set in Windows as being associated with this file type, you can just open a **.sqlplan** file and see the full graphical plan in SSMS, including having all the popups working.

## 6.4 Saving and sharing deadlock graphs

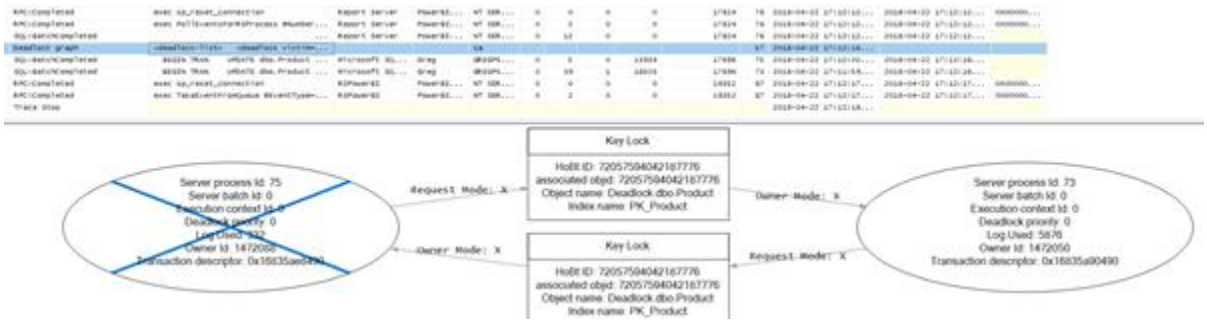
In an earlier post, I described how query plans could be saved as **.sqlplan** file, shared, and loaded again in SQL Server Management Studio (SSMS). It's also possible to extract them out of SQL Server Profiler or Extended Events Profiler.

This is useful, but the same applies to deadlock graphs. SQL Server 2005 added **Deadlock graph** as a type of event in SQL Server Profiler. (It's also part of Extended Events Profiler).

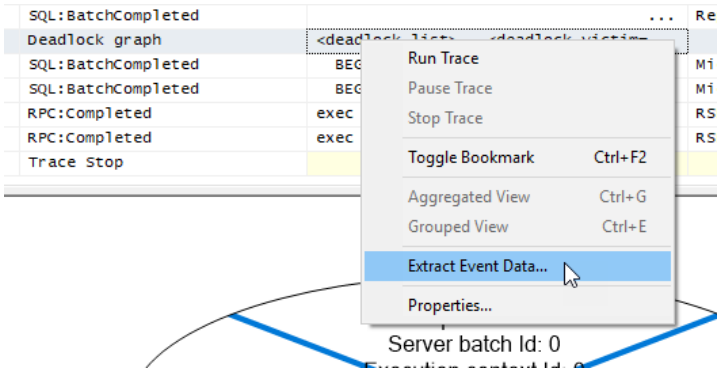
If I open a new trace in Profiler, I can add Deadlock graph to the list of events:



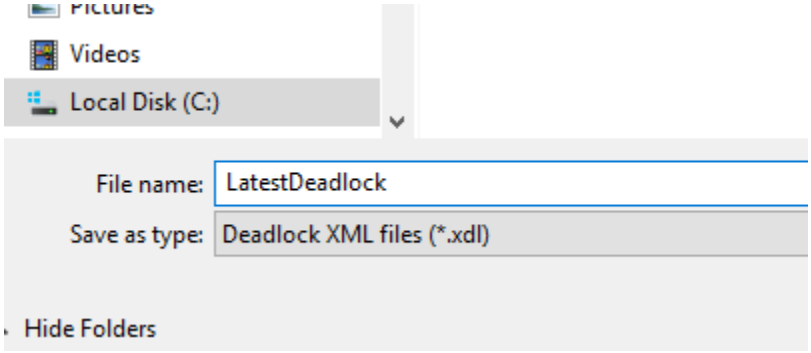

Then when a deadlock occurs, we can see an entry for it in the trace:



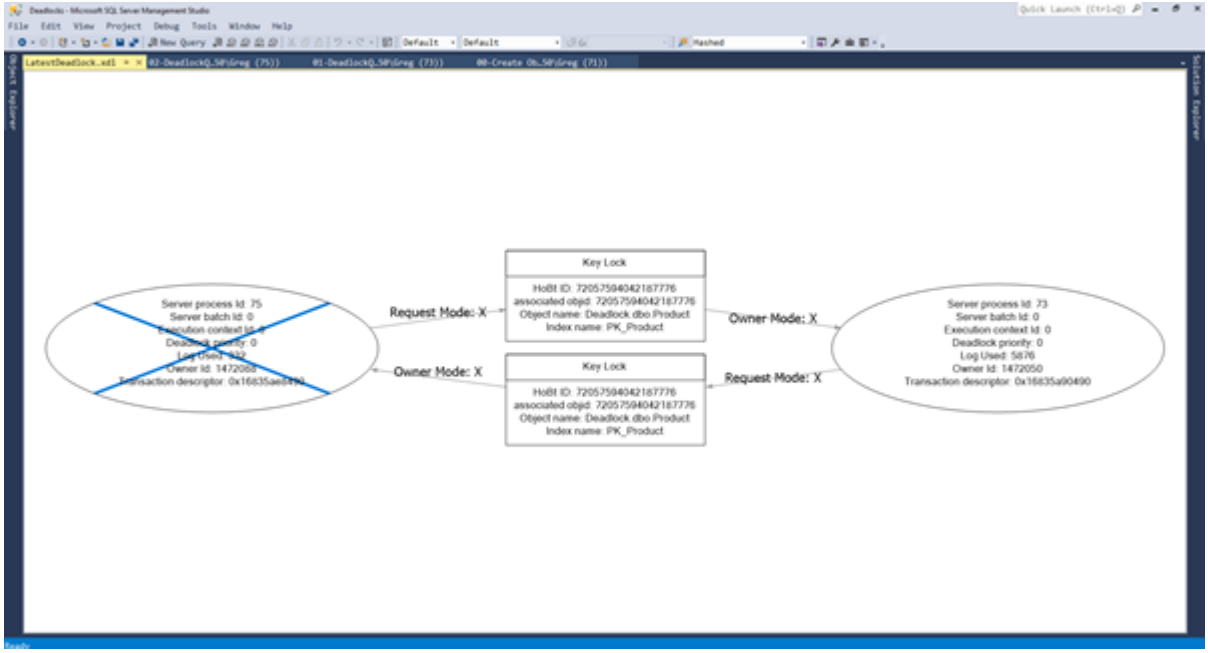
Now if you want to share that with someone else, you can right-click the deadlock graph event and extract the event data:



When you do this for a deadlock, the output is saved as a **.xdl** file extension.



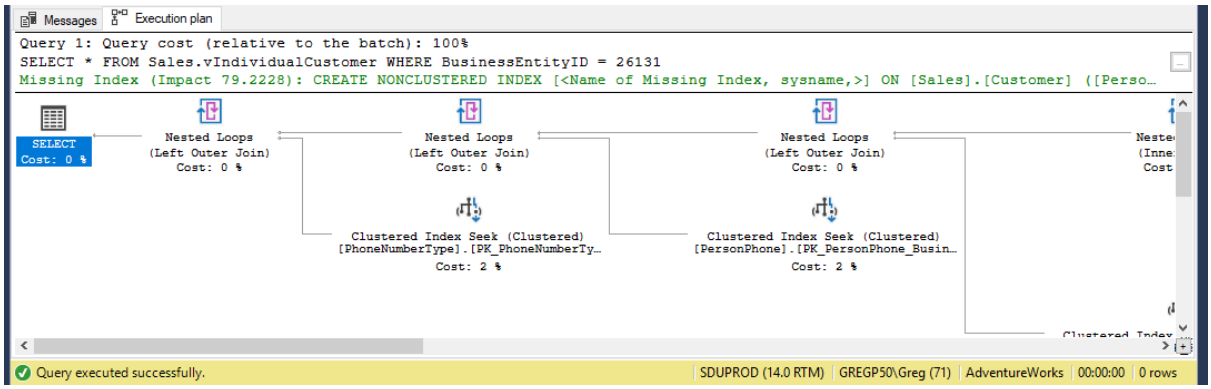

You can then send it to someone else and they can load it in SSMS:



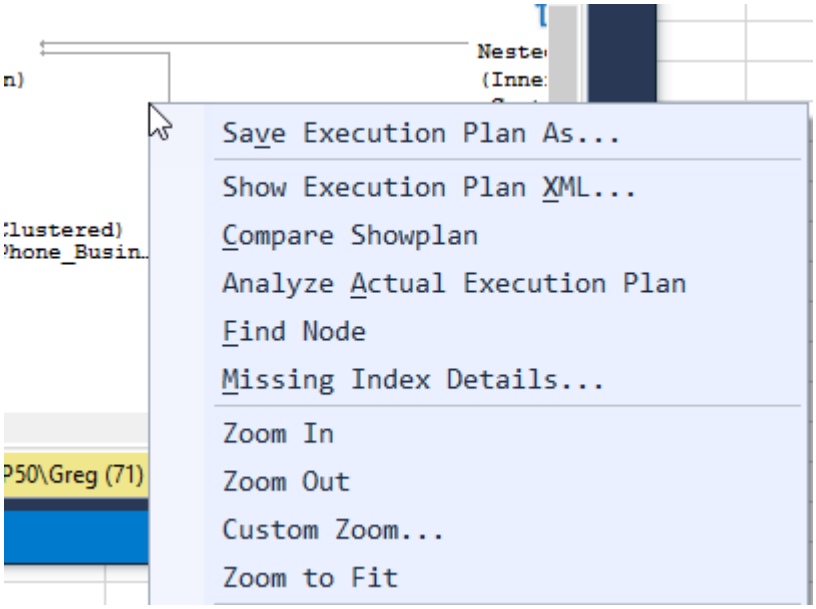

One important thing to note is that a deadlock graph is actually a list of deadlocks, not just one deadlock. When a deadlock graph is opened in SSMS, it only shows the first deadlock in the graph.

## 6.5 Zooming and navigate execution plans

SQL Server execution plans can become quite large. That makes them hard to navigate because you are endlessly scrolling around the results pane in SQL Server Management Studio (SSMS).

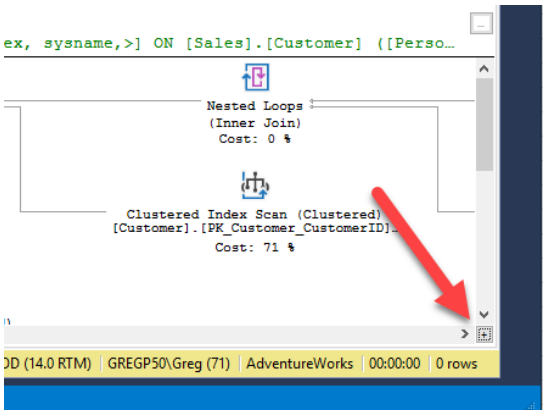



The pane does have some zoom features. Note that if I right-click in the whitespace, I get these options:

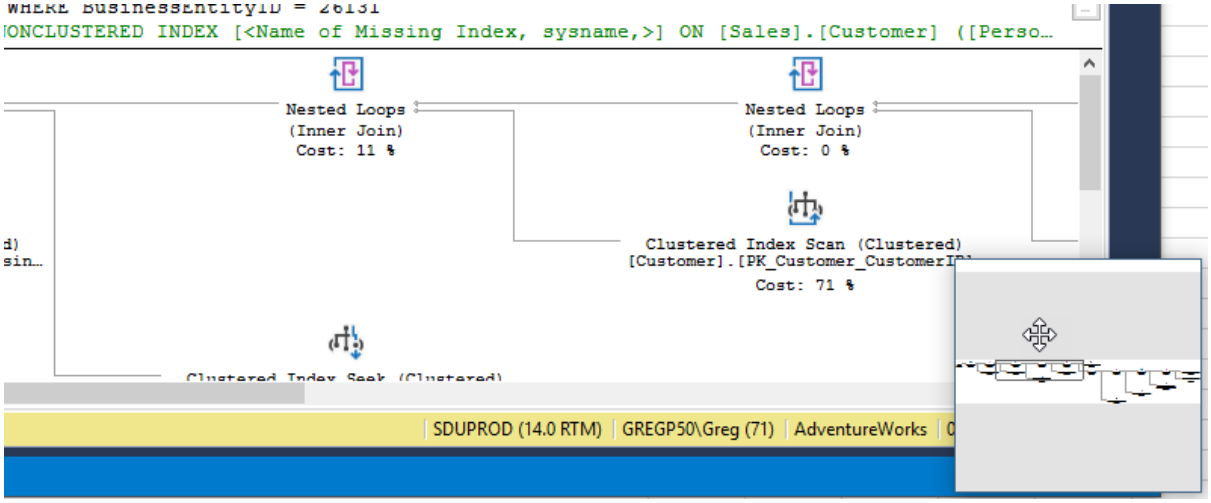



So I can zoom in and out, set a custom zoom level, or zoom until the entire plan fits. Generally though, that would make the plan too small to read, as soon as you have a complicated plan.

But in **one of the least discoverable UI features in SSMS**, there is an option to pan around the plan.



If you click and hold that little + sign, you'll find you can pan around within the plan:




That's very nice but I think it needs to be a little easier to find.

# 7 Projects/Solutions

## 7.1 Change default text in new query window

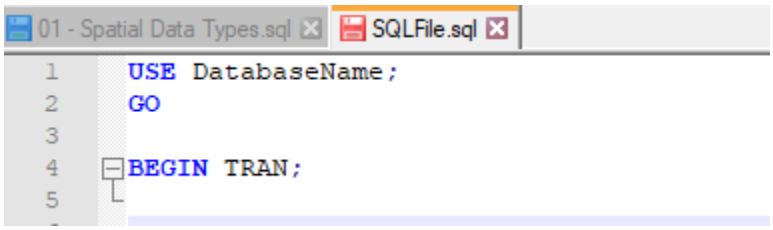
In SQL Server Management Studio (SSMS), when you click New Query, a new query window is opened, but because it's blank, what might not be immediately obvious is that it's based on a template.

The location of the template depends upon the version, but for SSMS 17.6, you'll find it in this folder:

**C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Binn\ManagementStudio\SqlWorkbenchProjectItems\Sql**

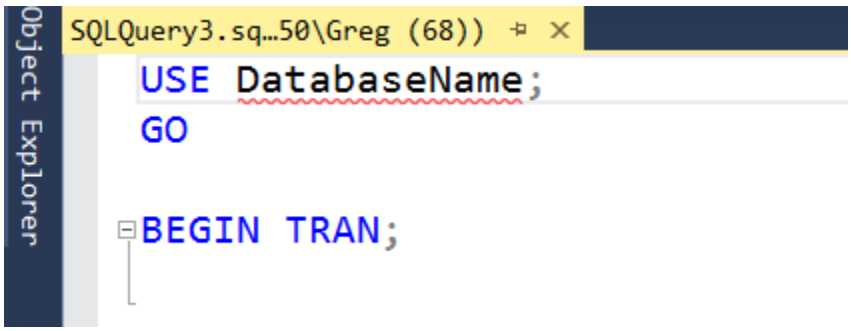
The file is called **SQLFile.sql**.

One of the things that I often forget to do is to change my connection to the correct database. Let's add a USE statement to make that obvious.




I've also put a BEGIN TRAN that I might use before doing any ad-hoc modifications.

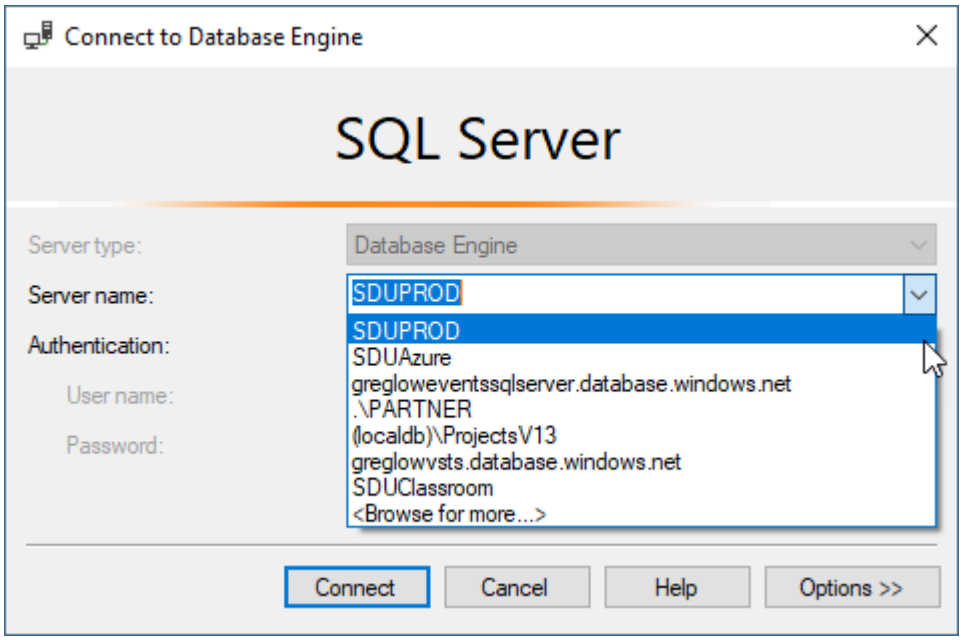
So I save that file, and when I do my next **New Query** in SSMS, I see this:




If I was really keen, I might also add templated values to be completed.

## 7.2 Clear server list in SSMS

SQL Server Management Studio (SSMS) keeps a list of the server names that you have connected to, and prompts you with those when you drop-down the list while making a connection:

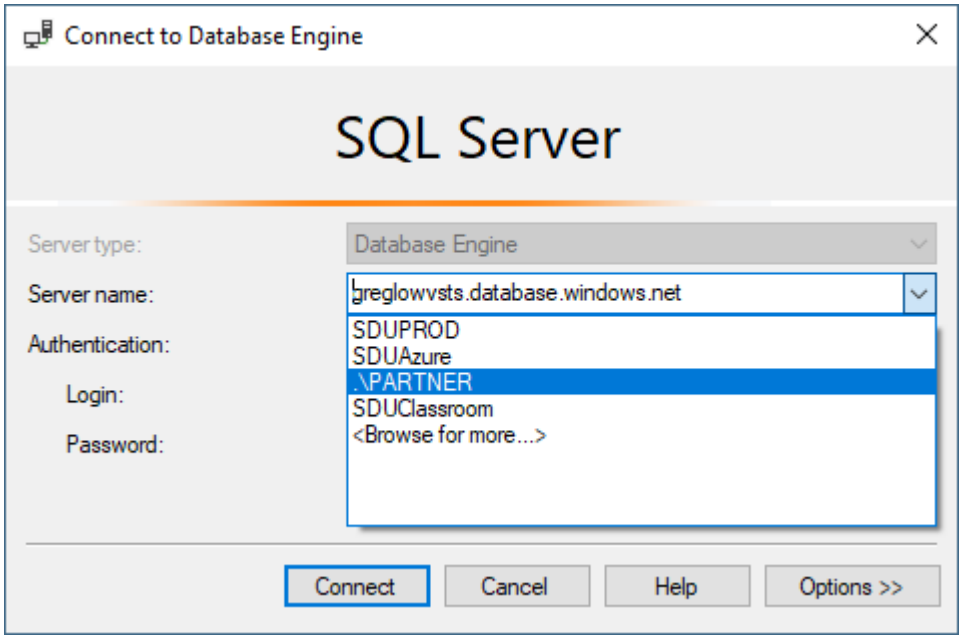



Eventually, that list can either become messy, it can include servers that don't exist anymore, and so on. You might want to clear up the list.

To do this in early versions of SSMS, you needed to locate the SqlStudio.bin file from the Documents and Settings area in your user profile.

Fortunately, that's no longer required.

All you need to do is to open this dialog, arrow down to the ones that you want to remove, and hit the Delete key.

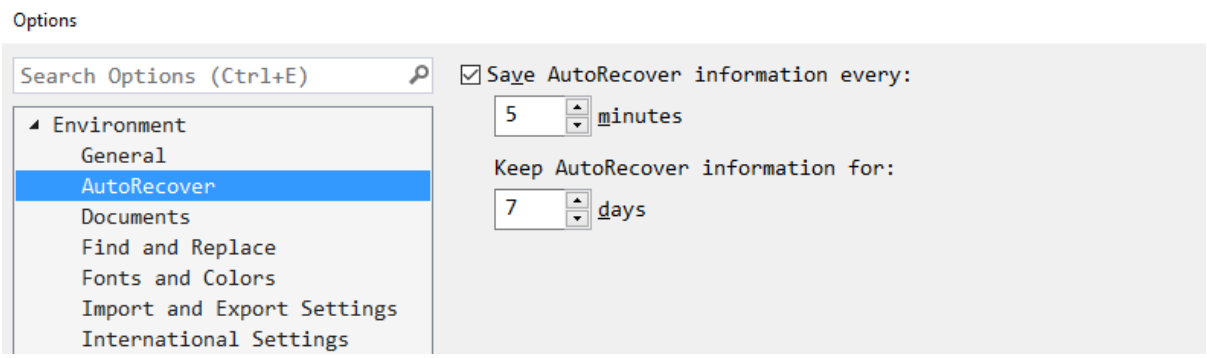


## 7.3 Configure autorecover time, and recover unsaved queries

Every now and again, I come back to my laptop and find that it has rebooted for some reason, while I wasn't expecting it. A prime cause of that is Windows Updates. I really, really wish that wasn't so, but someone at Microsoft has decided that I must apply these updates. I have very little control over the time when that occurs. For example, if I'm on the road delivering presentations, there's no "wait till I get home" option for Windows Updates.

Either way, it's really helpful that SQL Server Management Studio (SSMS) now creates periodic backups of unsaved queries.

You can control how often it does this, and how long it keeps the files for by adjusting the values in Tools, Options here:



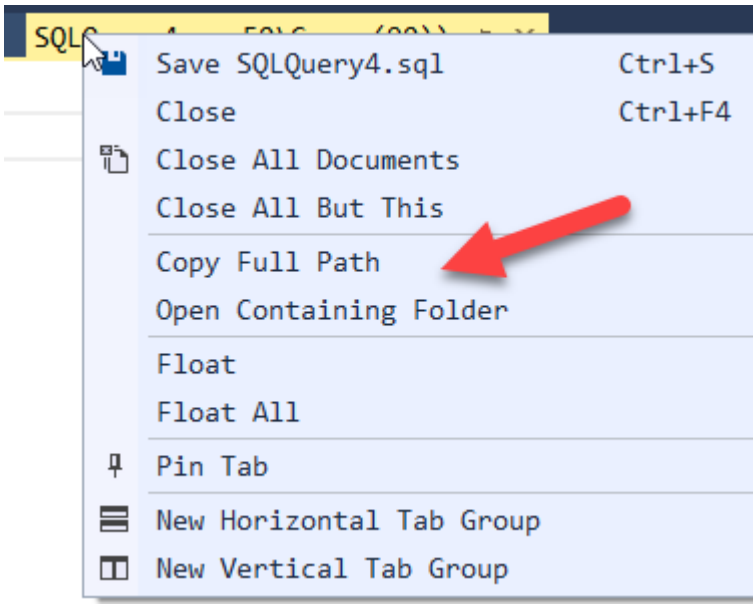

If SSMS crashes, you'll be prompted to recover the unsaved files when it restarts. But if you want to just go and find these files after another type of shutdown that you weren't planning on, you'll find them in your Documents folder in Windows, under the subfolders, SQL Server Management Studio, then Backup Files, then the name of a solution (likely Solution1 if you didn't create a scripts project).

## 7.4 Accessing script files and folders in SSMS

This one is a very simple and quick tip.

When working in SQL Server Management Studio (SSMS), I often need to open File Explorer in the folder where the script file is stored. Turns out there is an easy way to do that.

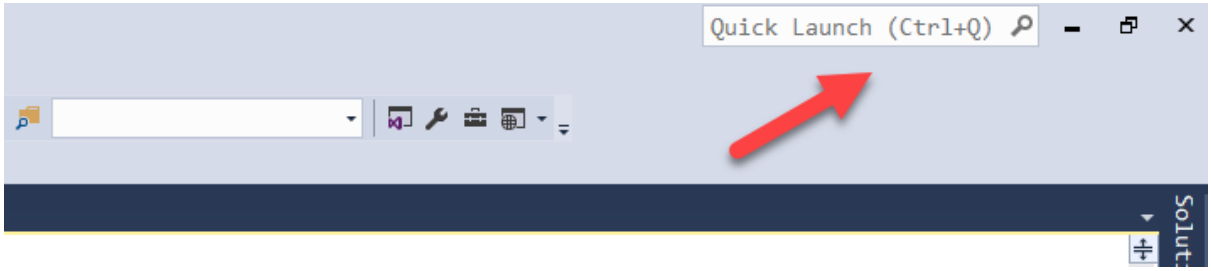
There are two interesting options when you right-click the tab at the top of a query window:




Note that you can open the containing folder for the script. You can also copy the path to the script into the clipboard.

## 7.5 Using quick launch

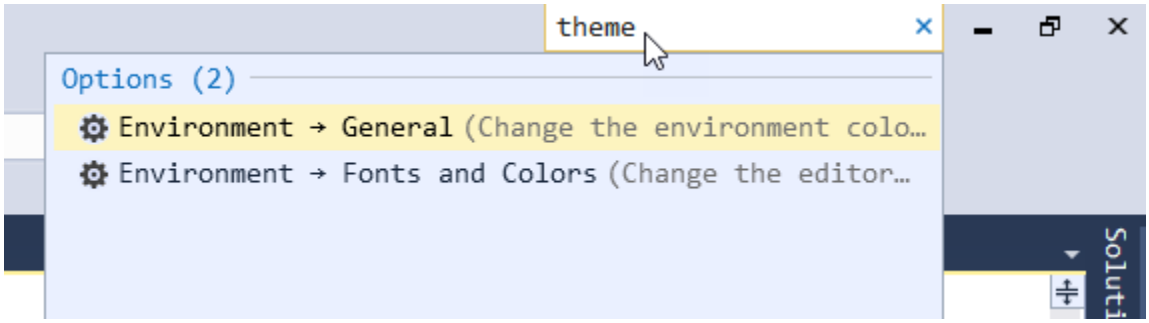
In SQL Server Management Studio (SSMS) for SQL Server 2016, a new search tool called **Quick Launch** was added. It's this bar up the top:




Note that there's another bar underneath it to the left. That's not the Quick Launch bar. That's the **Find** bar.

While the Find bar is useful for searching for text within queries, etc., the Quick Launch bar is useful for searching within SSMS itself.

This is great because it means you don't have to remember where all the options for various things are set. Here's an example:




I typed theme and it told me about relevant things. The first is spot on, and is where I can change the theme, and the second is relevant too. Perhaps it's just the fonts that I want to change.

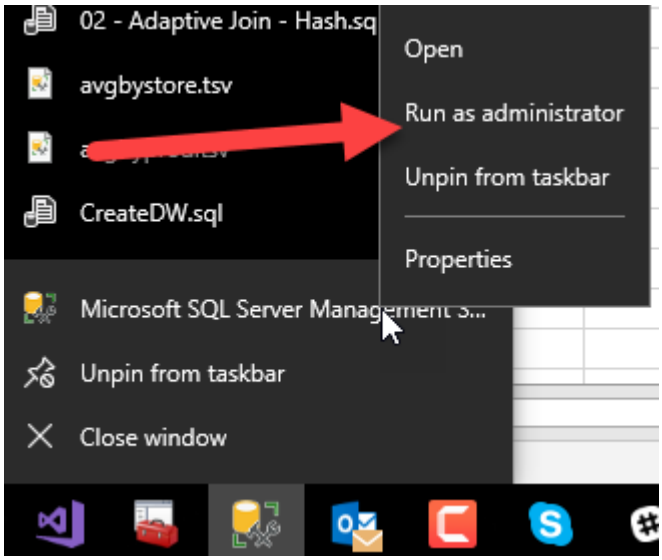
## 7.6 Run SSMS as someone else

You don't always want to run SQL Server Management Studio (SSMS) as your current login for Windows.

Now if all you want to do is to use a SQL Server login, then that's easy. When you connect to a server in Object Explorer, or when you start a new Database Engine query, you can just choose SQL authentication instead.

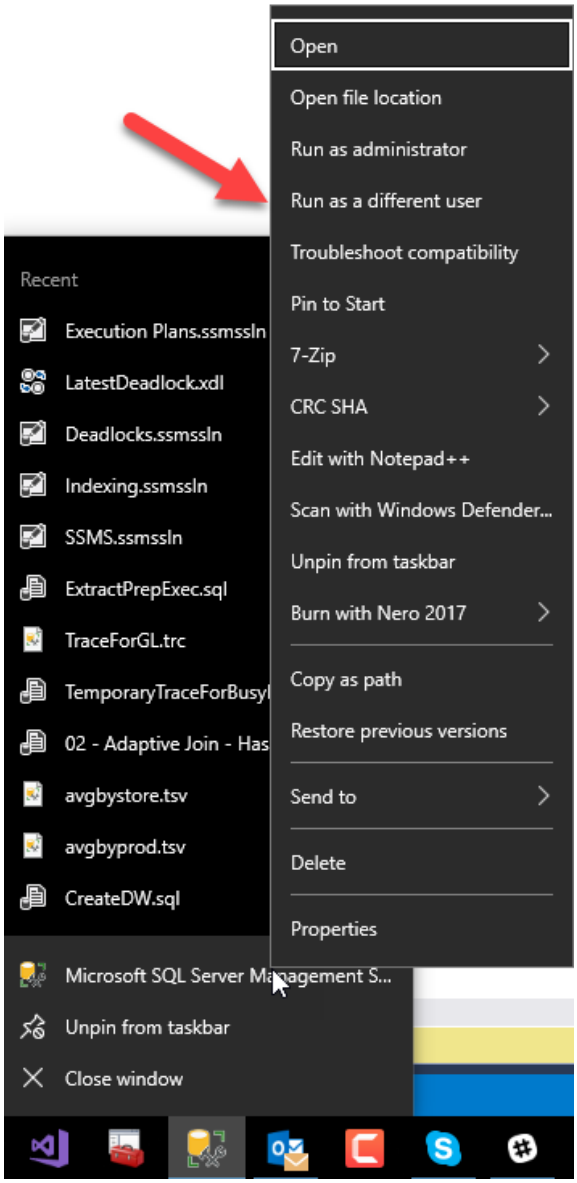
But three other scenarios commonly occur.

If you need to run SSMS as an administrator on a machine with UAC, you can do this:




You right-click the link to SSMS and choose **Run as administrator**.

Another option is that you need to run as another user ie: not just an administrator. If you are using Windows 10, the way to do this is to hold the Shift key before you right-click the SSMS link. Then you see this instead:




And you can then choose the Run as a different user option to log on as someone else just for this one application.

The third scenario is when you are needing to make a Windows authenticated connection to a server, but from a system that is not part of the domain. In that case, you need to run SSMS from the command line like this:

**runas /netonly /user:targetdomain\targetusername "C:\Program Files (x86)\Microsoft SQL Server\140\Tools\Binn\VSShell\Common7\IDE\Ssms.exe"**

You will need to modify that command to point to the current location of Ssms.exe for the version that you are running.

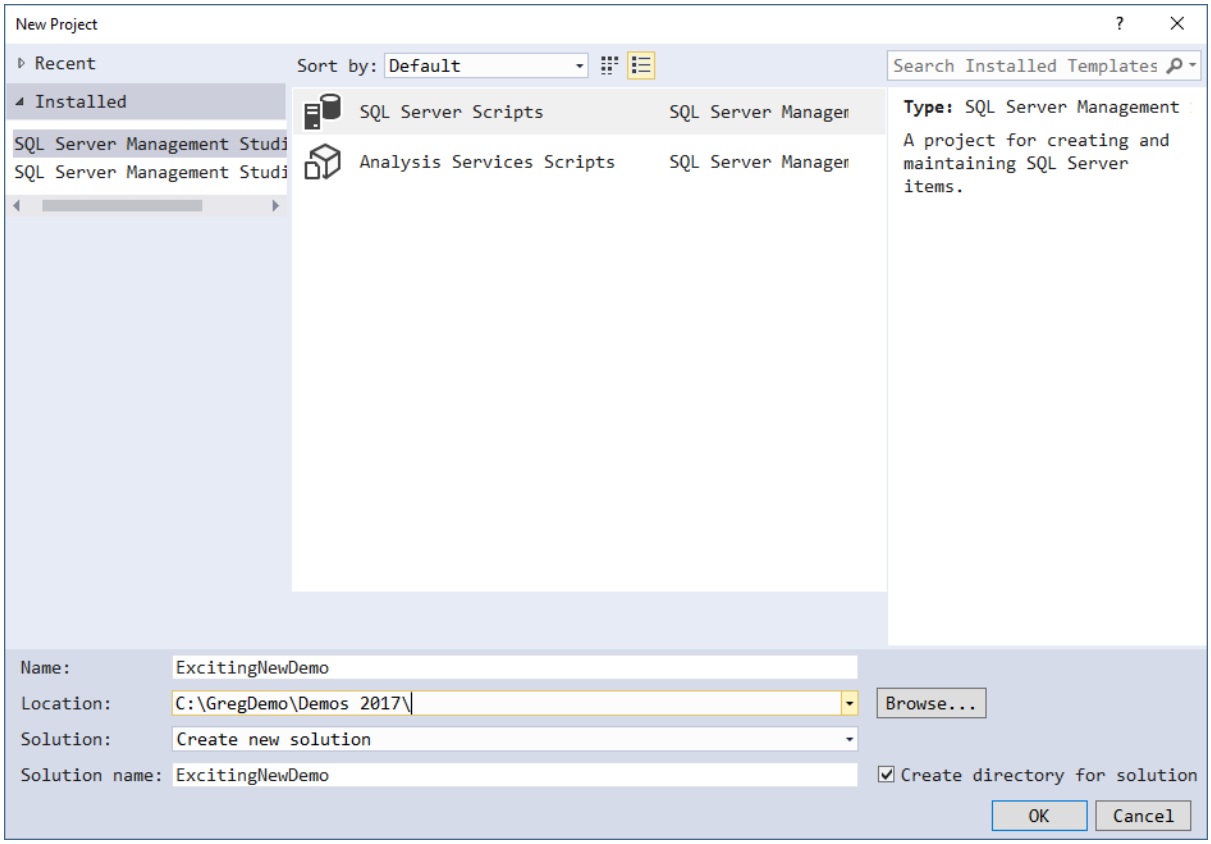
You will then be prompted to log on to the domain.

## 7.7 Script projects and solutions

I'm puzzled that so few people use script projects and solutions when working with SQL Server Management Studio (SSMS).

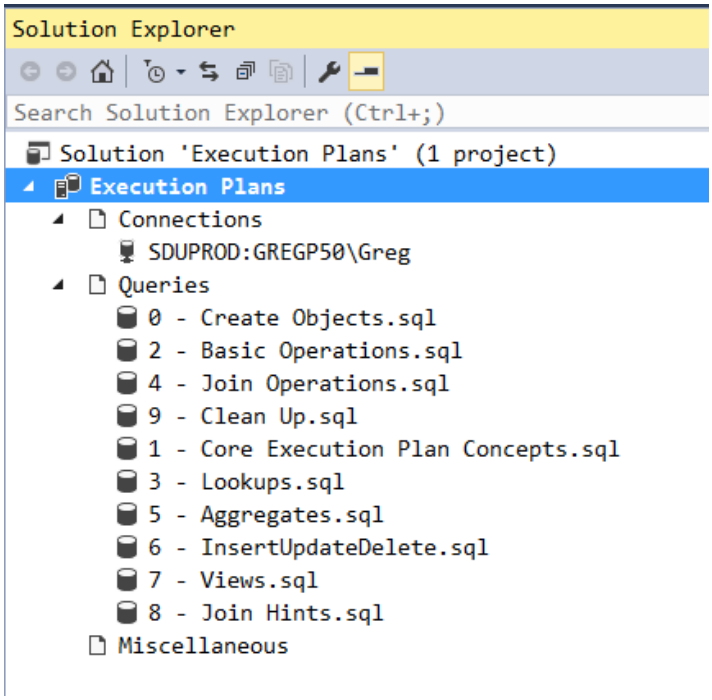
They are easy to use. Let's see an example:

Instead of just starting to create scripts, from the File menu, click New, then Project. You are greeted with the new Project dialog which also allows you to create a solution.




I've selected SQL Server Scripts as the project template. Note there is also one for Analysis Services scripts. I've named the project, picked a location, and chosen to create a new solution. I might choose to create a solution with a different name if it will contain multiple projects. In this case, I'm not doing that.

To get to the point faster, here's one that I created earlier:

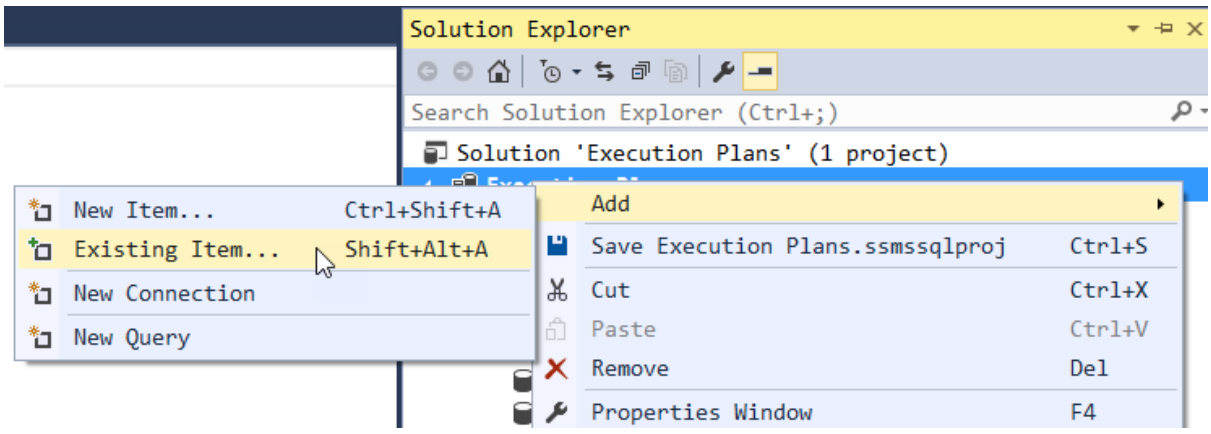



There are three sections of interest:

**Connections** – keeps details of all connections used by the project. I typically only have one for any project but I can imagine scenarios where I would have more. A big advantage of this is that if I need to change the connection, I do it once, and all scripts here will use it when I open them.

**Queries** – as the name says. A nice addition in recent versions is that it automagically keeps them shown in alphabetical order. It didn't used to do that so we had to remove them all and put them back in.

**Miscellaneous** – you'll find if you right-click this that there's nothing of interest. So how does something get there I hear you ask? Well it's when you right-click the project and ask to add an existing item.




If you pick any file apart from those with a **.sql** extension (such as a text file), they'll be placed into the Miscellaneous folder.

These projects are a great free built-in way to manage your scripts.

## 7.8 Start faster by disabling certificate revocation checking in constrained environments

If you have ever started SQL Server Management Studio in an isolated environment (ie: one with no external Internet connectivity), you'll find that it's slower to start.

That's because SQL Server uses signed assemblies, and whenever an application with signed assemblies starts, it needs to check whether or not the certificate that they were signed with has been revoked. It's not good enough to just check if it's a valid certificate.

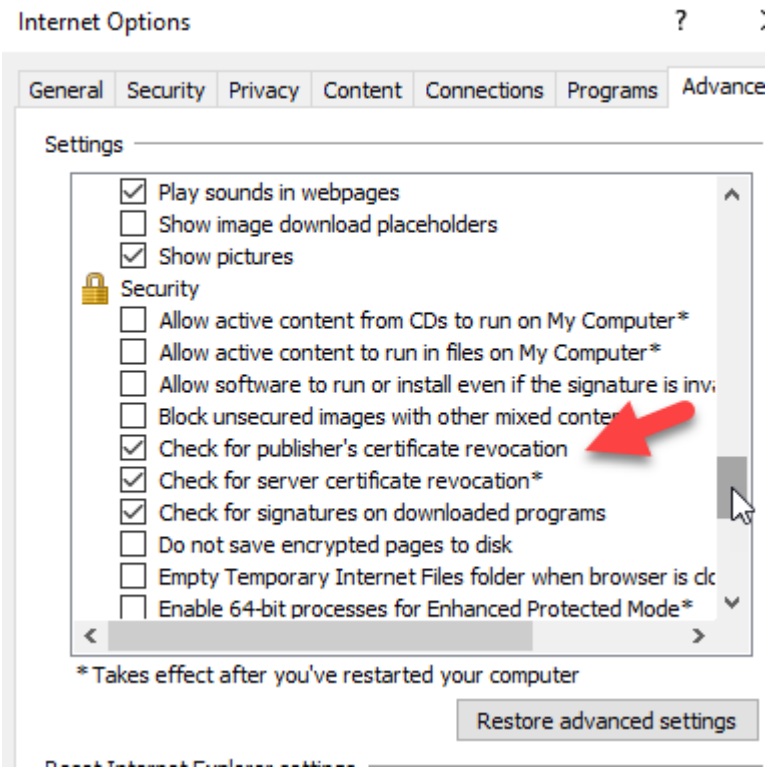
Certificates include a CRL (Certificate Revocation List) and this tells an application that's trusting the certificate where to check for a list of revoked certificates.

The problem is that when you try to locate a server in an isolated environment, you might see a delay of around 40 seconds as the DNS timeout occurs.

**If you have an environment like this, you might decide that it's safe to turn off this revocation checking. That's a call you need to make, and if in doubt or don't understand the issues, leave it on.**

I often run across this though as I have isolated virtual machines running in Hyper-V on my laptop. SSMS isn't going to be able to look these details up, nor is any other application running within the virtual machine.

Turning this off is a registry setting but can be done via your browser settings. For example, in Internet Explorer, look in Settings, Internet Options, then Advanced. Scroll down to find it:

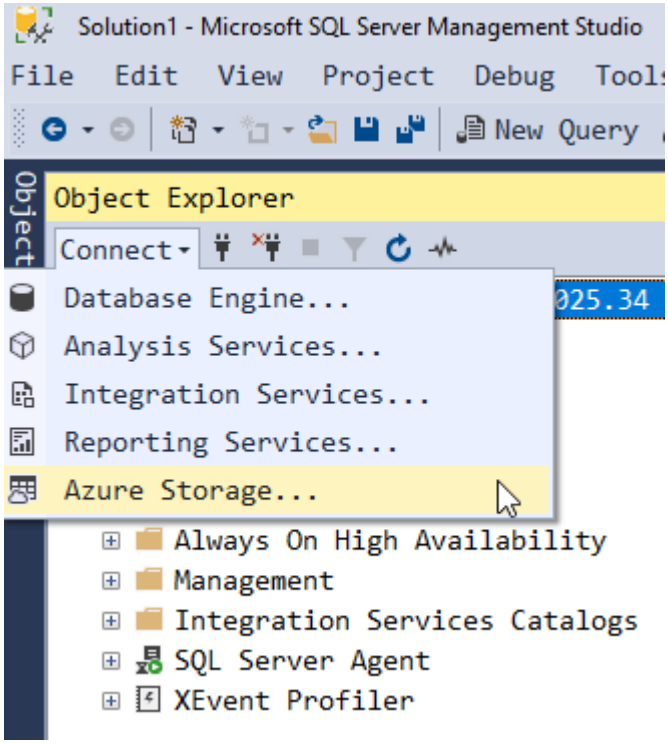



**Keep in mind that if you disable this, it applies to all checking of certificates on the machine. As I said, if in doubt, don't do it.**

## 7.9 Connecting to Azure Storage and other services

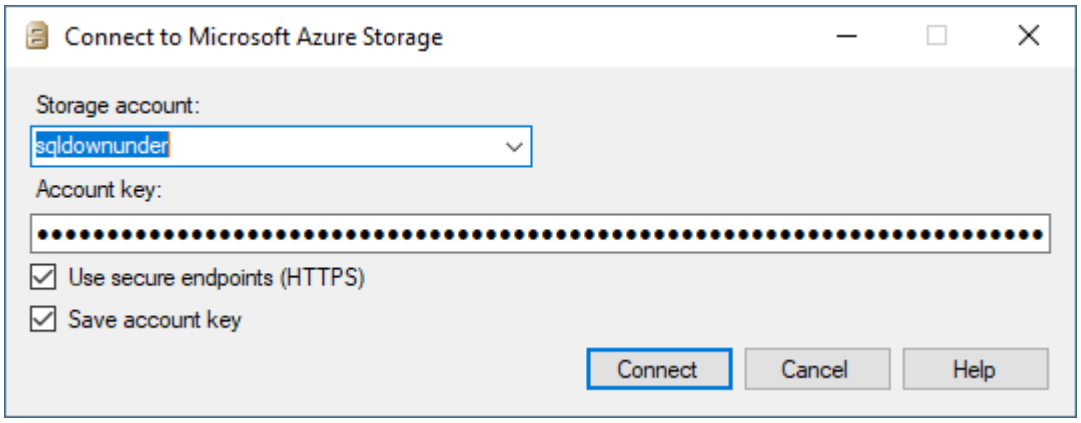
SQL Server Management Studio (SSMS) is a great tool for working with SQL Server relational databases but it can do much more than that.

In Object Explorer, note that you can easily connect to other types of services:

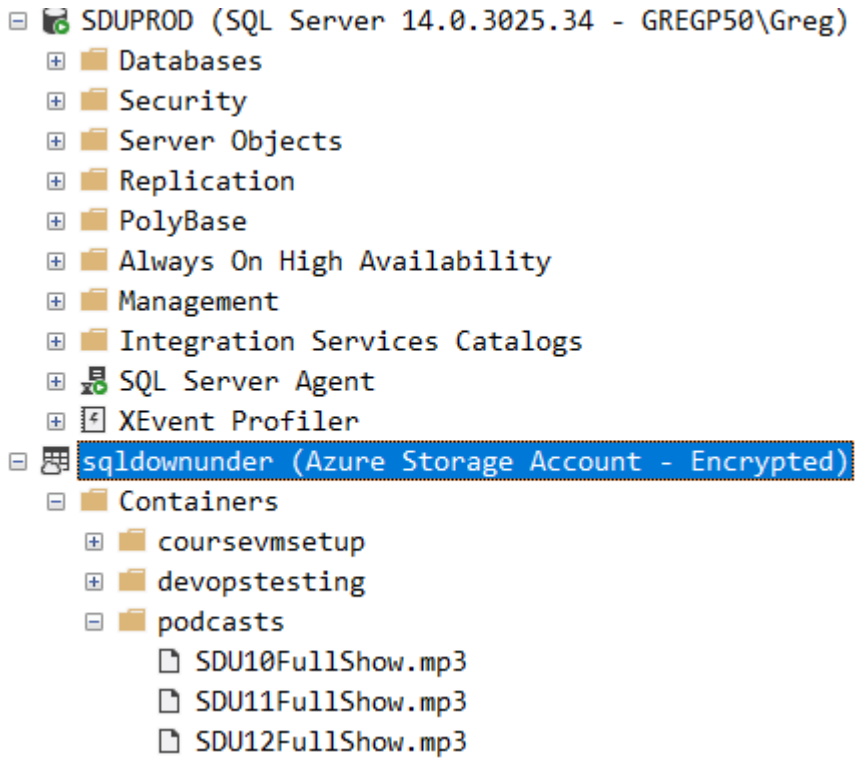



For a long time, it has been able to connect to Analysis Services to manage SSAS databases, both tabular and multi-dimensional. It can connect to Integration Services but that's to the older style interface for SSIS. Nowadays, you should use the SSIS Catalog instead. There are a few items that you can configure via the Reporting Services connection as well.

One option that is often quite unexpected though, is that you can connect to Azure Storage. Here's an example. After I click Azure Storage, I'm prompted for the storage account and the key. This can be the primary or secondary account key. (Note that it can't just be a shared access signature connection).



From there, once the connection is made, you can drill into the containers and their contents:




Generally, I would use **Azure Storage Explorer** to manipulate these storage accounts, but this option can be useful if you are using BACKUP TO URL and you need to just check the backup files that you are creating.

# 8 Window Tabs

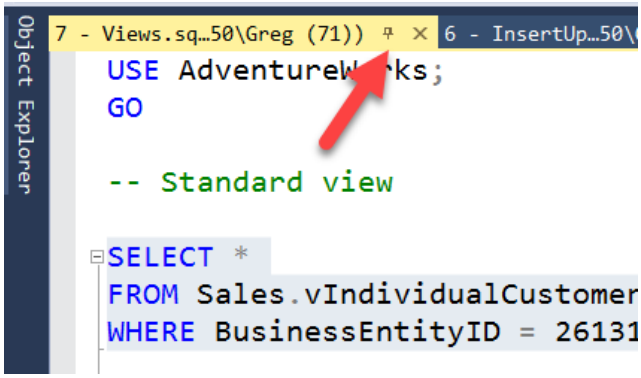
## 8.1 Pinned Tabs

When you get to a large number of query windows or other documents open as tabs in SQL Server Management Studio (SSMS), it can start to be difficult to keep track of them, and to find them when needed.

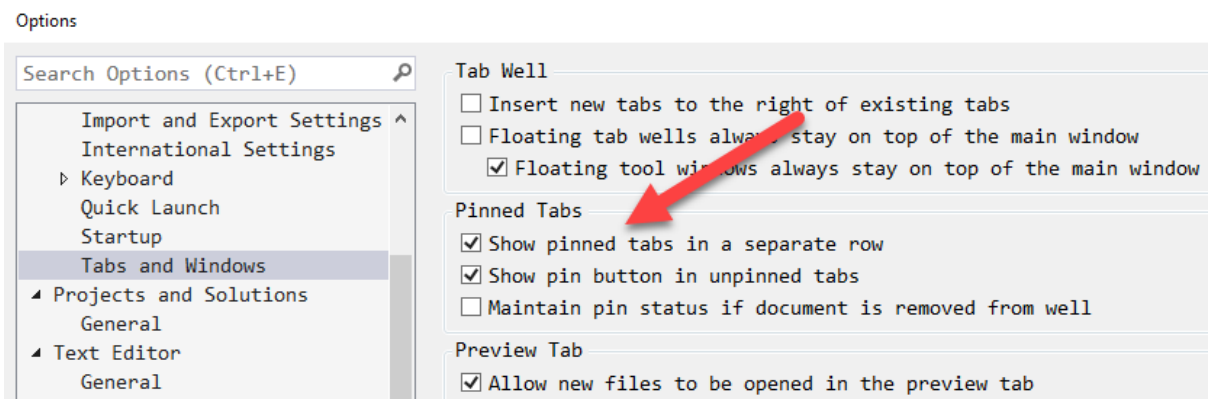
It's not too bad when you can immediately find the tab that you want in the drop-down list:


But if you have more tabs than are shown in this drop-down list or if, like me, you often end up with many of them without names (as they are temporary), it can get very hard to find the few that you are mainly referring to.

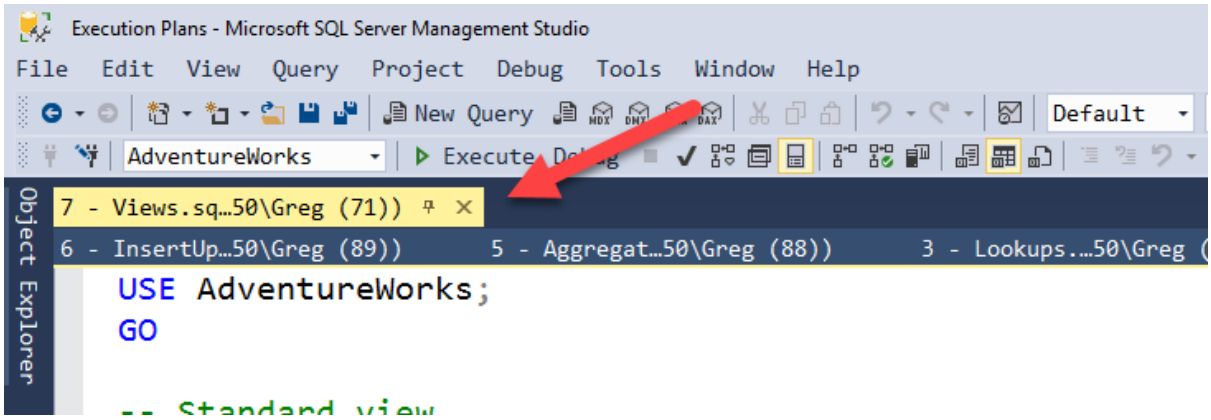
Just like you can with tool windows, you can pin tabs:




Once you do this, they stay against the left-hand side (by default). Now that's not bad but again if you have a few of them, there's another option that can help.



Once you configure that, another row of tabs appears in SSMS:



## 8.2 Reset window layout

One of the problems with applications that have highly-configurable user interfaces (UI) is that users can end up configuring them in ways they hadn't intended, and then don't know how to get back to where they were.

I remember the first time that I was at a session with a presenter from Microsoft showing the (at the time) new personalization options in ASP.NET. You could build a website and let the user determine how the site should be laid out, to suit themselves.

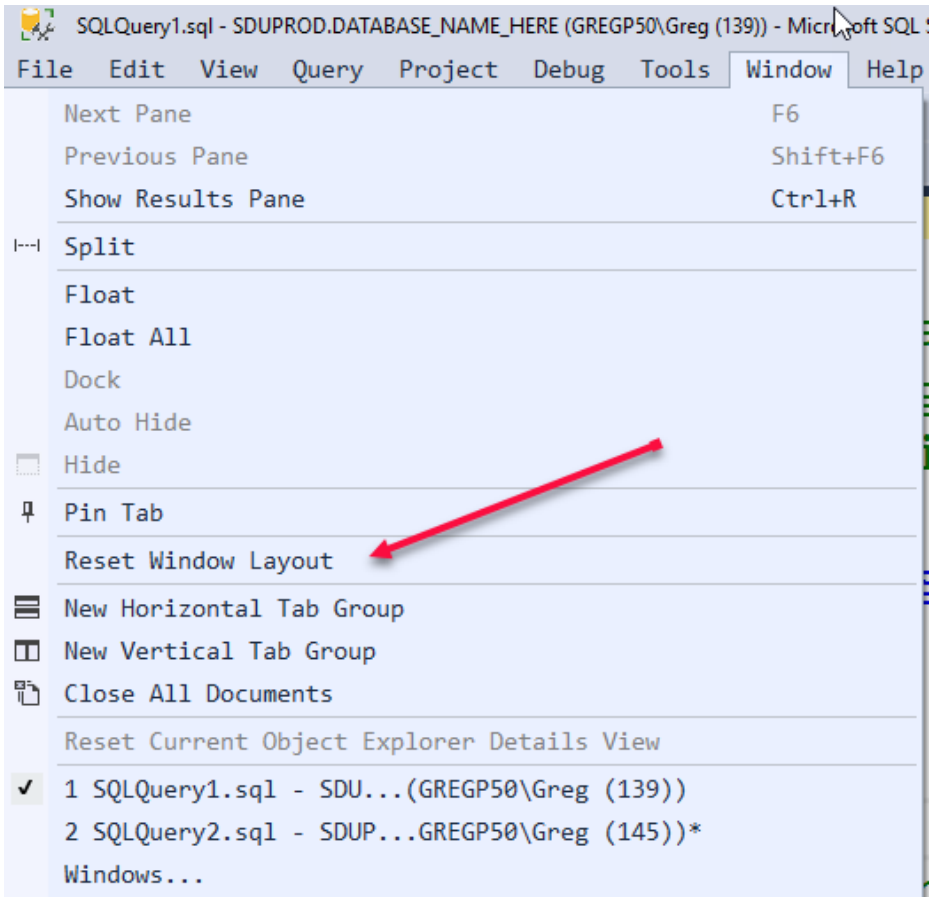
Overall, I can't say that I really like working with websites like that but I can understand the potential appeal. But I can easily see how end users could get really messed up.

I remember asking the presenter if there was a simple button that put the site back the way it was initially developed and removed the user's modifications, so that a user could always just get back to square one.

He told me "**ah no, there isn't an option like that**".

I'm glad that **@sqltoolguy** 's team that work on SQL Server Management Studio (SSMS) aren't part of that thinking. While SSMS is very configurable, I have seen people get really messed up with the window management in it. They ended up dragging a window when they meant to drag something else, or did another action that changed their UI and it stuck. Then they don't know how to "fix" it.

In SSMS, there's a wonderful option in the Window menu, that does just what's needed:

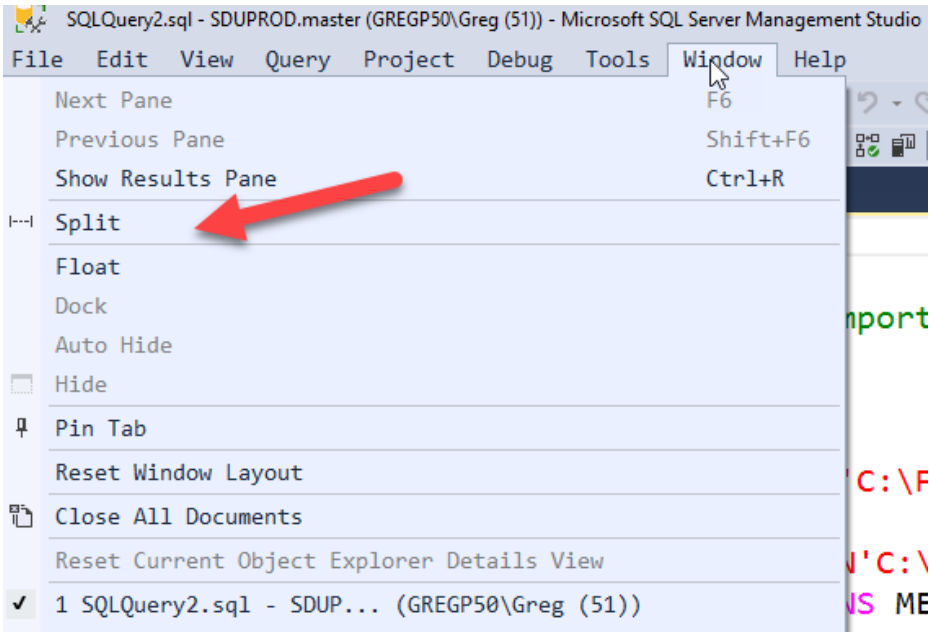



**Reset Window Layout** is the "get me back to where I was" menu item.

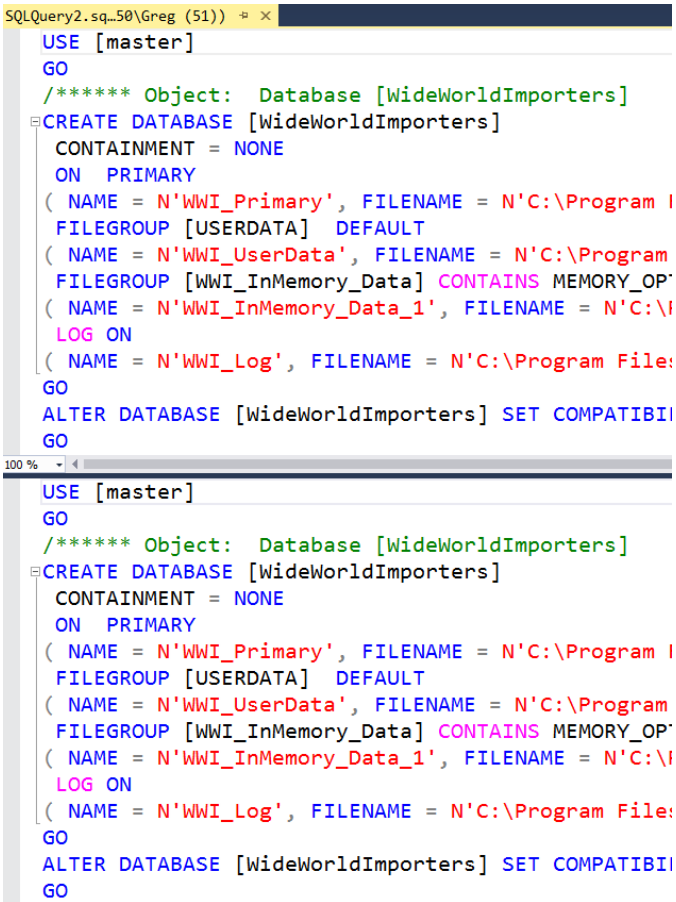
## 8.3 Split Screens

If you are working with really long script files in SQL Server Management Studio (SSMS), you might need to work on more than one part of the script at the same time. Perhaps you need to work on a function, and also on the code that calls the function.

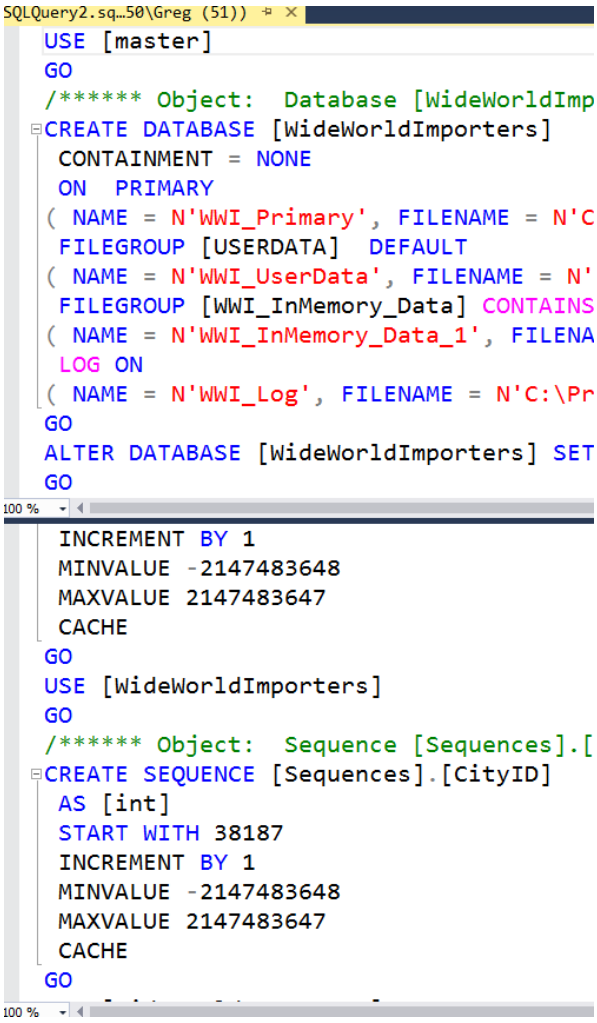
On the Window menu, there is a Split option.




When you first do this, you'll see a split window with the same query at top and bottom:



You can then scroll each vertically and resize them independently, and work on different parts of the same script:




The easiest way that I've found to close this, is to double-click on the dark bar in the middle, but there is also a Remove Split option in the Window menu.

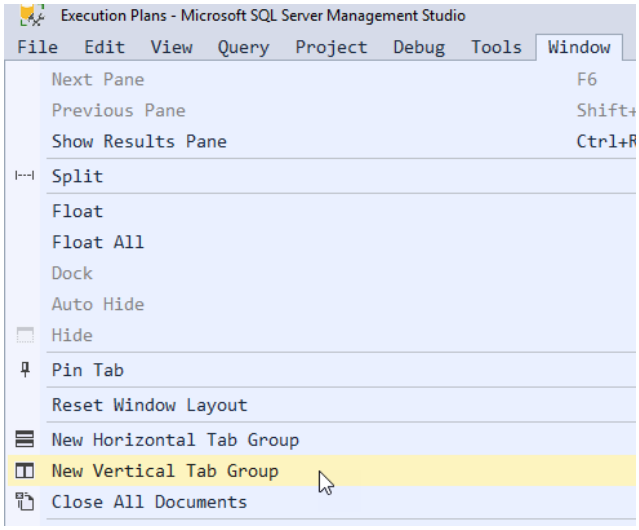
## 8.4 Tab Groups

In another section, I showed how you might use split windows to allow you to work on different parts of a single query at the same time.

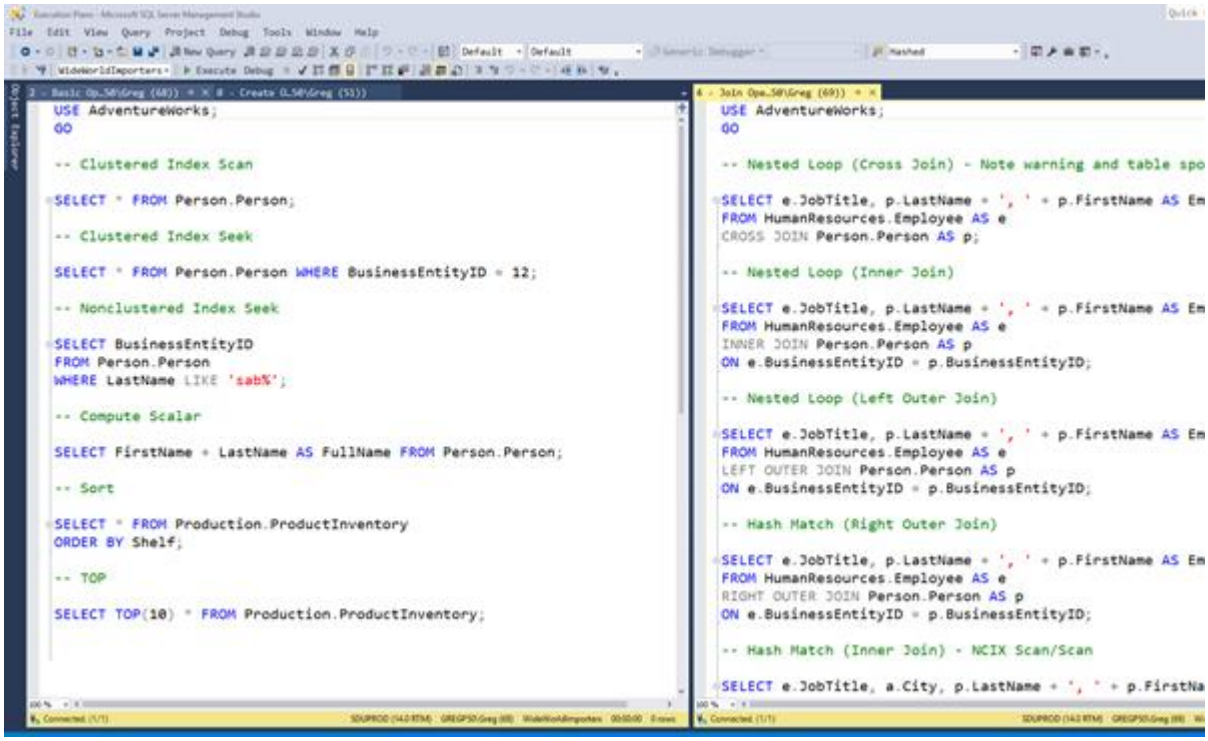
But what if you need to work on two queries and see parts of both of them?

That's where tab groups can help you. You can create both vertical and horizontal groups. For me, the most useful is typically side-by-side vertically, for when I'm comparing two sections of code.

From the Window menu, I choose New Vertical Tab Group:

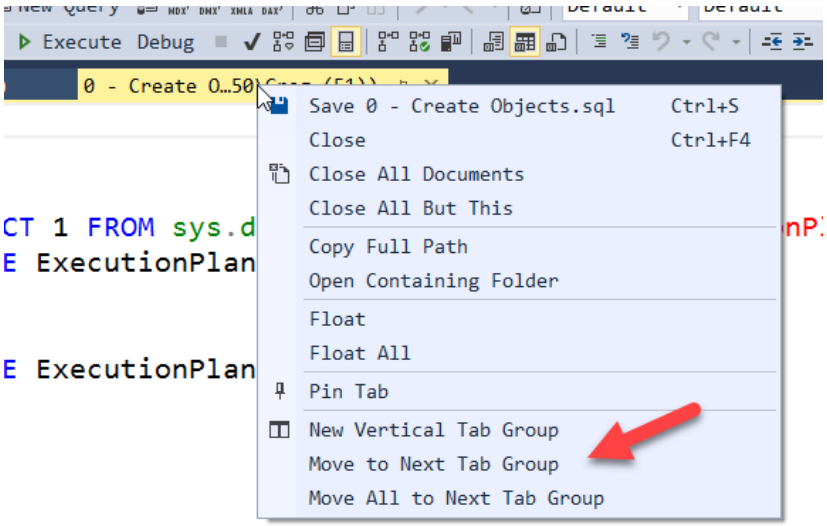



After I click that, as long as there are two or more queries open, I'll see this:




I then have two queries that I can work with side-by-side. A horizontal tab group places them one above the other.

In the image above, I had two queries in the left group, and one in the right group. You can also move queries between these groups. If I right-click the tab heading for one of the queries, I see this:



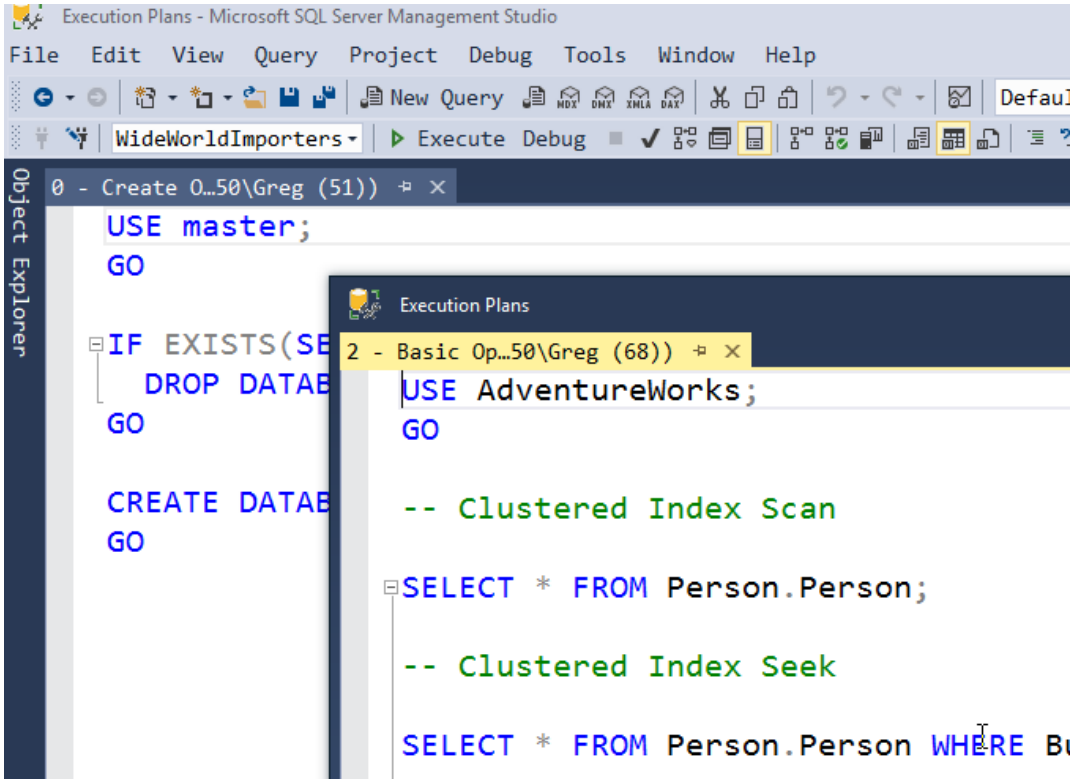

I can create yet another tab group based on this tab, or move it to the next tab group, or even close this tab group by moving all tabs within it to the next tab group.

## 8.5 Undock tabs and windows (including to other screens)

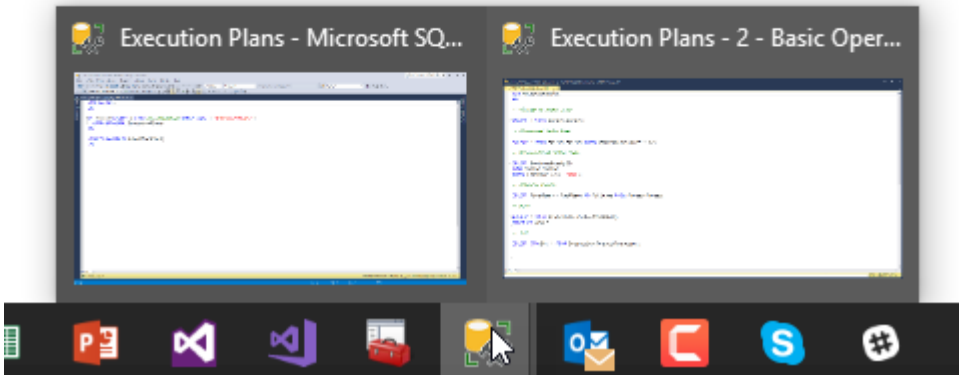
Like Visual Studio that it's based upon, SQL Server Management Studio (SSMS) is very flexible when working with query windows and tabs.

Most people realize that you can undock and move tabs and windows around. Usually they do that by accident and then realize that the **Reset Window Layout** option in the Window menu is helpful.

But one option I've found that many people don't seem to realize is that you can undock just a single query window and move it outside the bounds of SSMS. You can even place it across on another screen if you have multiple screens.




It then also appears separately in your taskbar in Windows:




You can later drag it back to redock it.

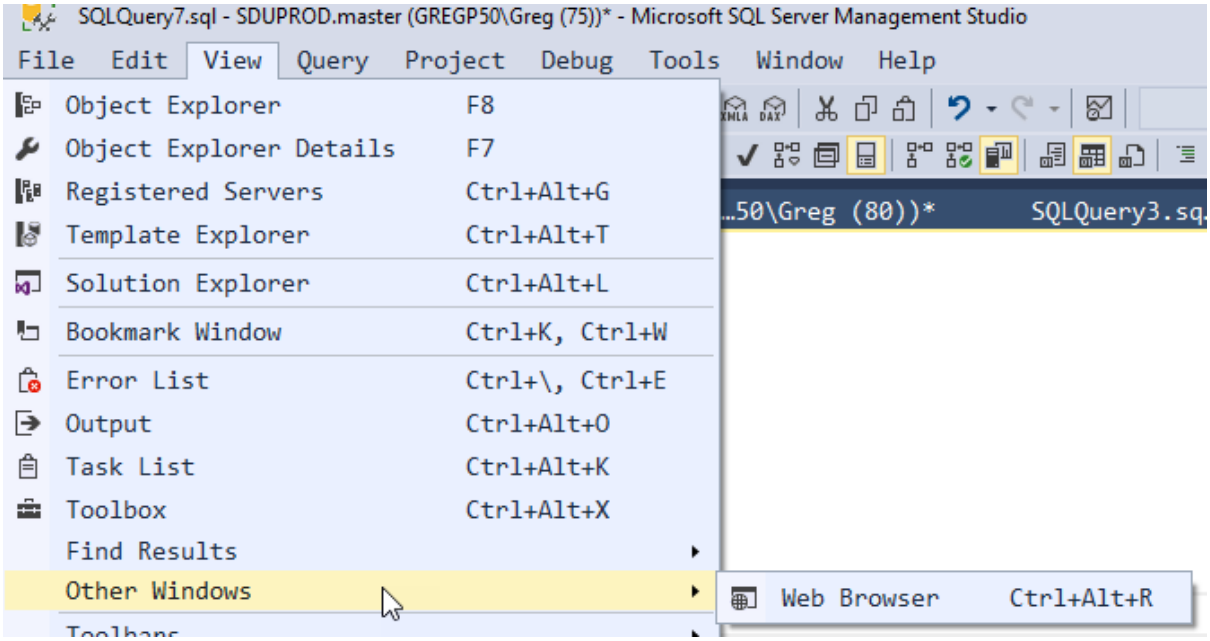
## 8.6 Built-In Web Browser

SQL Server Management Studio (SSMS) is a flexible tool. One thing that often surprises people is that it hosts a version of Microsoft Internet Explorer, right inside the application.

Why would SSMS have a web browser I hear you ask?

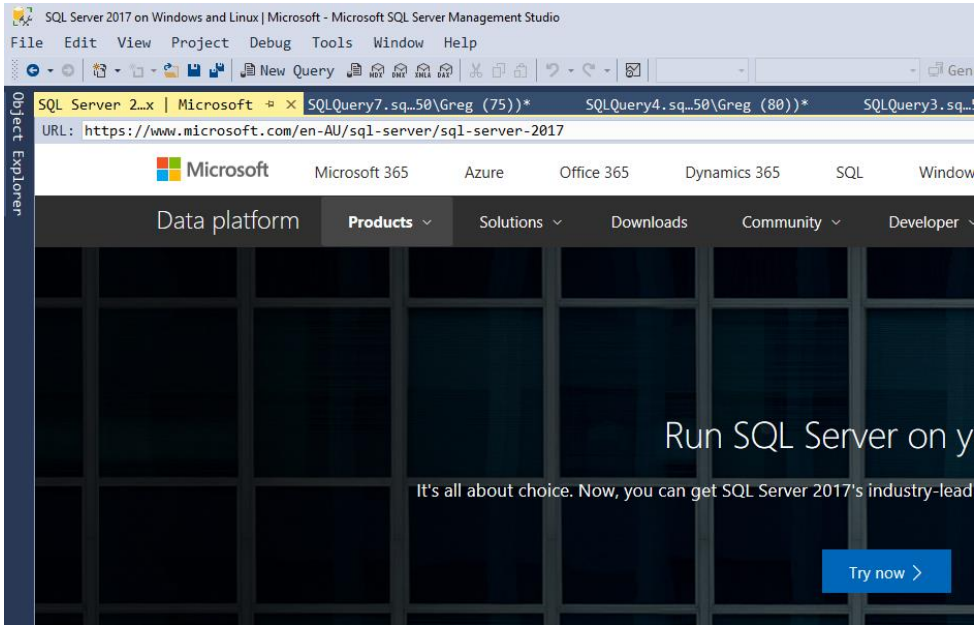
Well this web browser lets browse URLs, and reference links, without leaving the tool.

You can open it directly by choosing **Web Browser** from the **View** menu:




On my machine, it's under the Other Windows section. In fact, it's the only window there on my machine, which makes you wonder why it has a separate section in the first place.

It opens like any other tab:



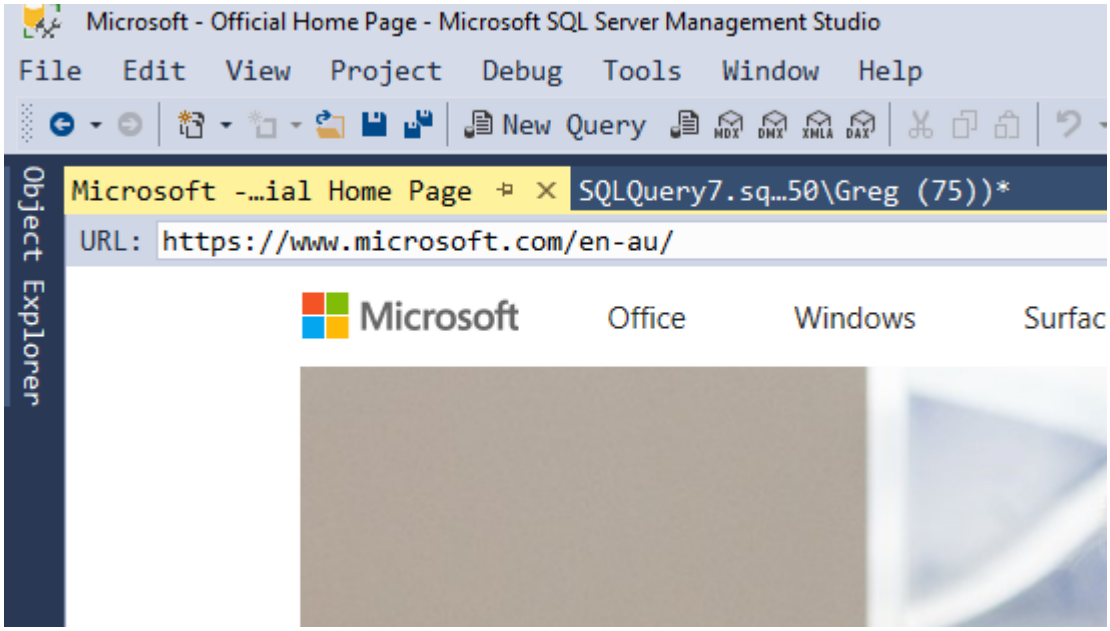
In SSMS query tabs, you'll notice that there is an auto-detection of URLs and auto-formatting/linking of them:


If I execute that as a query, it doesn't get hyperlinked in the results:


Or on the Messages tab:


If you hover over the URL within the query though, you'll see that you can Ctrl-Click to open it:

If you use Ctrl-Click the link, it will open in the embedded web browser:




I'm hoping there will be an option to get a much more up-to-date browser though as support for Internet Explorer is waning everywhere.