SQL Fast Track

Sample manual - first two chapters

Wise Owl Training

TABLE OF CONTENTS (1 of 9)

TABLE OF CONTENTS (2 of 9)

TABLE OF CONTENTS (3 of 9)

TABLE OF CONTENTS (4 of 9)

TABLE OF CONTENTS (5 of 9)

TABLE OF CONTENTS (6 of 9)

TABLE OF CONTENTS (7 of 9)

TABLE OF CONTENTS (8 of 9)

TABLE OF CONTENTS (9 of 9)

CHAPTER 1 - DESIGNING DATABASES

The world runs on relational databases. If you understand the principles upon which these are built, you'll find it much easier to write SQL to get information out of them!

*This manual gives an overview only of database design principles. If you want to delve deeper, try Googling phrases like **Third Normal Form**, **Database Normalisation** or **Entity Diagram**. If nothing else, this will give you an impressive search history in your browser!*

1.1 The Four Stages of Database Design

There are four stages to designing a relational database, shown below (using the example of creating a simple database to hold films; or movies, if you must).

Stage 1 – Deciding what to Include

A good way to do this is to create a spreadsheet of the data you want to include for each film:

Title	Oscars	Director	Date of birth	Studio
Armageddon	0	Michael Bay	17/02/1965	Touchstone Pictures
Bad Boys	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Bad Boys II	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Dead Poets Society	1	Peter Weir	21/08/1944	Touchstone Pictures
Master and Commander ...	2	Peter Weir	21/08/1944	20th Century Fox
Pearl Harbor	1	Michael Bay	17/02/1965	Touchstone Pictures
The Rock	0	Michael Bay	17/02/1965	Hollywood Pictures
The Truman Show	0	Peter Weir	21/08/1944	Scott Rudin Productions

We want to assign each film to a director, but we don't want to have to type each director's name in over and over again!

The aim of designing a relational database is to ensure that you don't hold information twice:

Title	Oscars	Director
Dead Poets Society	1	Peter Weir
Master and Commander ..	2	Peter Wier
The Truman Show	0	Peter Weird

Not only is holding duplicate information inefficient, but it also means that spelling mistakes will creep in. Here listing out films directed by **Peter Weir** would miss out the last two films, as his name has been misspelt.

Stage 2 – Dividing Data into Tables

Having decided what data you want to include, the next stage of database design is to decide which table each bit of information belongs to:

Title	Oscars	Director	Date of birth	Studio
Armageddon	0	Michael Bay	17/02/1965	Touchstone Pictures
Bad Boys	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Bad Boys II	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Dead Poets Society	1	Peter Weir	21/08/1944	Touchstone Pictures
Master and Commander ...	2	Peter Weir	21/08/1944	20th Century Fox
Pearl Harbor	1	Michael Bay	17/02/1965	Touchstone Pictures
The Rock	0	Michael Bay	17/02/1965	Hollywood Pictures
The Truman Show	0	Peter Weir	21/08/1944	Scott Rudin Productions

These are all details to do with the film itself.

These are to do with the director (name / birthday).

These are details to do with the studio.

There's no magic wand to make this easier, other than bitter experience of getting it wrong and having to start again! A good guideline is that if you find yourself typing in something twice, it probably belonged in a different table.

For our example above, there are clearly 3 separate entities: films, the directors who made them and the studios which produced them. Here are the fields that each table could contain:

Table	Fields
Film	**Title** and **Oscars Won**, plus something to identify which director and which studio made it
Director	**Director name** and **Date of birth**, plus some unique identifier for the director
Studio	**Studio name**, plus some unique identifier for the studio

What you need to do next is to decide what form these unique identifiers should take.

Stage 3 – Choosing a Primary Key for each Table

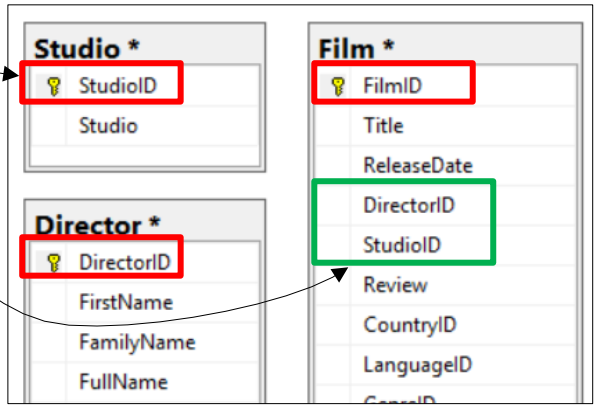
The *primary key* for a table is a field which tells you exactly which record you're considering (for example, if you know a film's **DirectorID** you can look up all of the director's other details).

From the above definition, it follows that two records in a table can't have the same value for the primary key field – the field is unique.

For our example, we could use the director and studio names as our primary keys, but SQL Server works most efficiently if the primary key is as short as possible, so we'll create new fields instead:

The primary key for each table is a number which uniquely identifies which studio, film or director we're looking at.

Each film contains fields specifying who directed the film, and which studio made it.



Here's what **The Sound of Music** would now look like:

FilmId	Title	OscarWins	DirectorID	StudioID
638	The Sound of Music	5	89	4

Including the director's unique number allows us to look up all their other details:

DirectorID	FullName	DoB	Gender
89	Robert Wise	10/09/1914	Male

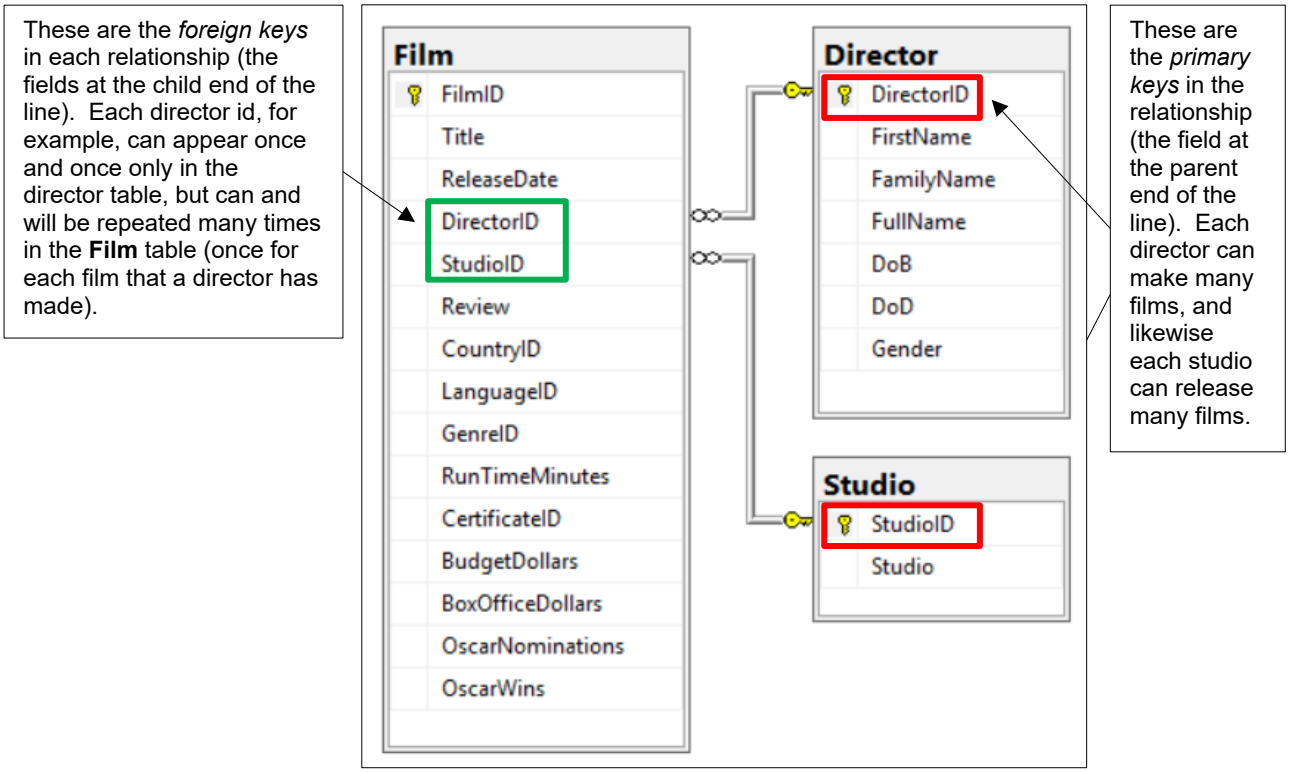
Including the studio's unique number allows us to look up its name:

StudioID	Studio
4	20th Century Fox

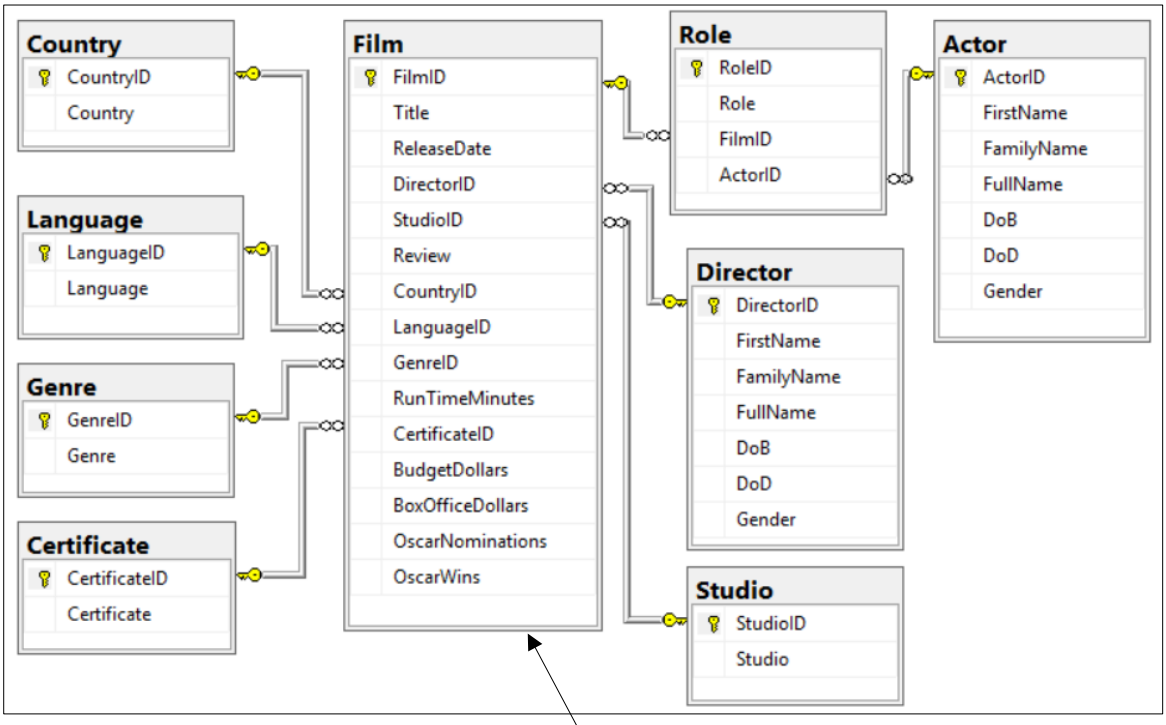
*If you're beginning to think that relational databases are just like a lot of spreadsheets joined together with a more efficient version of a **VLOOKUP** formula in Excel, you're absolutely right!*

Stage 4 – Creating Relationships and a Database Diagram

The last step in designing a database is to decide for each relationship that you create whether it is *one-to-many* or *many-to-one* (*parent-child* or *child-parent*):



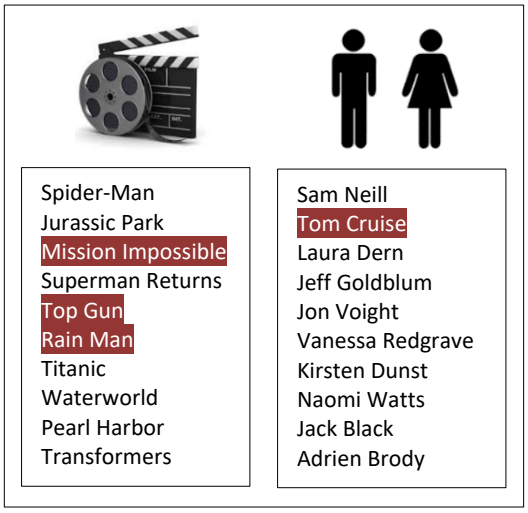
Database diagrams often involve hundreds of tables:



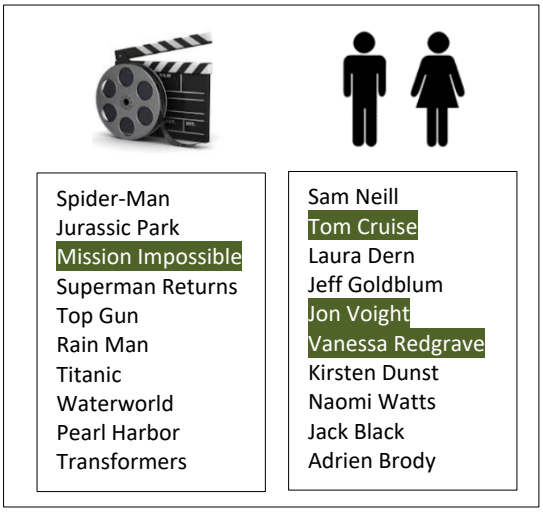
The **Movies** database we'll use in this courseware contains just 9 tables, and hence is untypically simple.

1.2 Many-to-Many Relationships

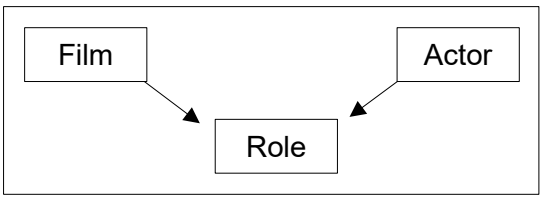
There's no such thing as a many-to-many relationship in SQL Server, but they do exist in real life:



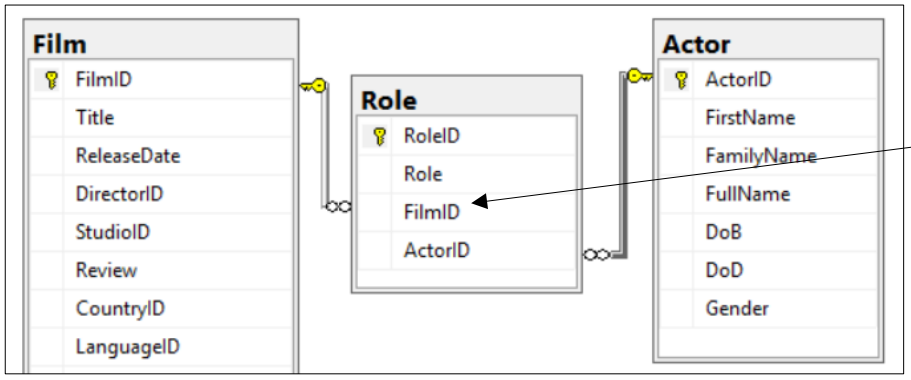
***Tom Cruise** has appeared in lots of films, but equally **Mission: Impossible** has lots of actors in it.*



The solution to this problem is to create a table that is a child to both of the two parent tables, as here:



Here's what the database would look like:



The **Role** table links the **Film** and **Actor** tables. Each film can contain many roles (otherwise a film could only have a single actor), but likewise each actor can have many roles (otherwise they would never work again after completing their first film).

Here are 3 rows from the **Role** table:

RoleID	Role	FilmID	ActorID
1	Ray Ferrier	33	1
2	Dr. Alan Grant	1	2
3	Dr. Ellie Sattler	1	3
202	Nathan Algren	41	1

Film number 1 appears twice in this list, as does actor number 1.

Here are the films and actors who are represented by these rows of data (the duplicate film name was **Jurassic Park**, and duplicated actor turns out to be **Tom Cruise**).

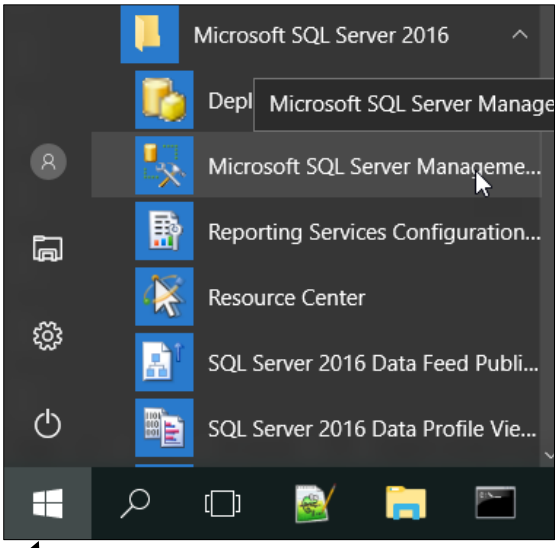
RoleID	Role	FilmID	ActorID	Title	FullName
1	Ray Ferrier	33	1	War of the Worlds	Tom Cruise
2	Dr. Alan Grant	1	2	Jurassic Park	Sam Neill
3	Dr. Ellie Sattler	1	3	Jurassic Park	Laura Dern
202	Nathan Algren	41	1	The Last Samurai	Tom Cruise

CHAPTER 2 - SQL SERVER MANAGEMENT STUDIO

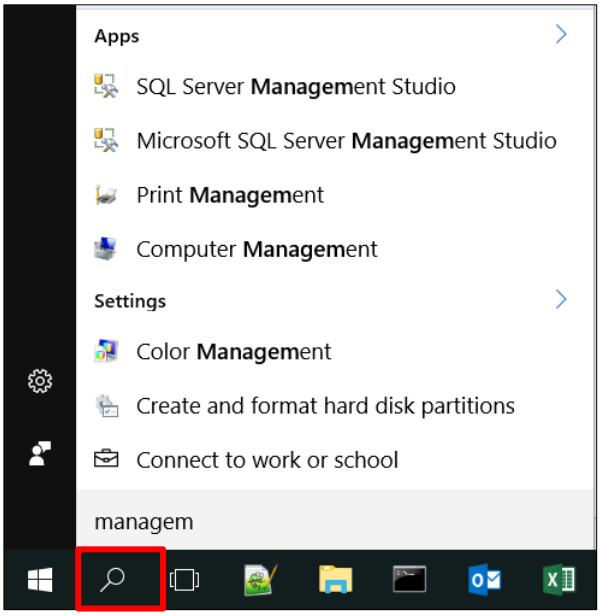
If you're writing SQL to get information out of a database created using SQL Server, the chances are that you'll use *SSMS* (*SQL Server Management Studio*) as your authoring tool.

2.1 Starting to Use Management Studio

You can start SSMS like any other application – here are a couple of ways using Windows 10:



Click on the Windows icon, and choose the program that you want to run …

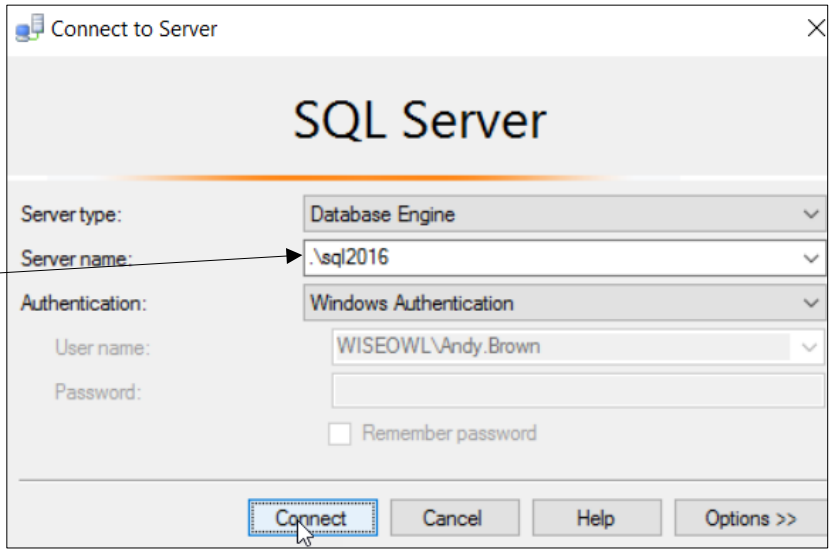


… or click on this tool and type in part of the program name (here typing **managem** has been enough to bring up the program name in the list).

You can then choose a database to use (or *connect to*):

You will see the SQL Server logo on screen while it loads:

You can then choose from the dropdown list which of your company's servers you want to connect to (your IT people should be able to advise on which to choose). If you use Windows authentication, you won't have to type in any more user names or passwords.



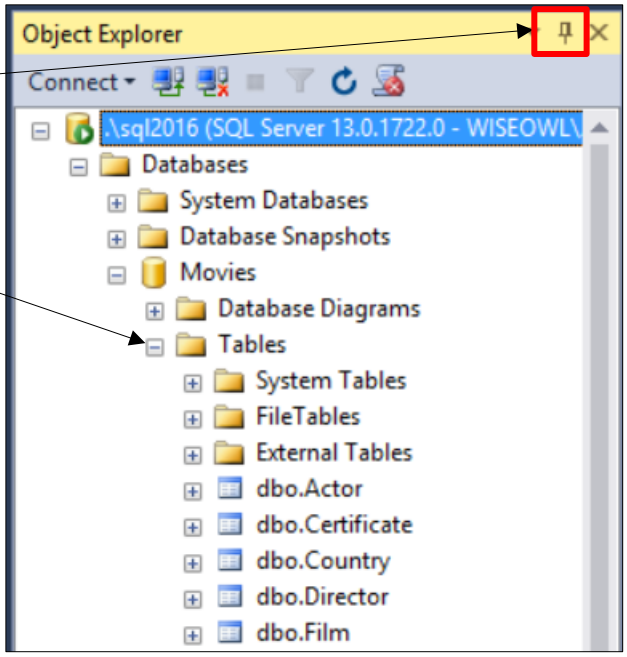
2.2 Object Explorer

When Management Studio loads, you should see the **Object Explorer** window (if it's not visible, press F8 to show it):

If you think Object Explorer is taking up too much room, click on this pin (it will go from vertical to horizontal, and the window will collapse until needed). Click again on the pin to make the window permanently visible again.

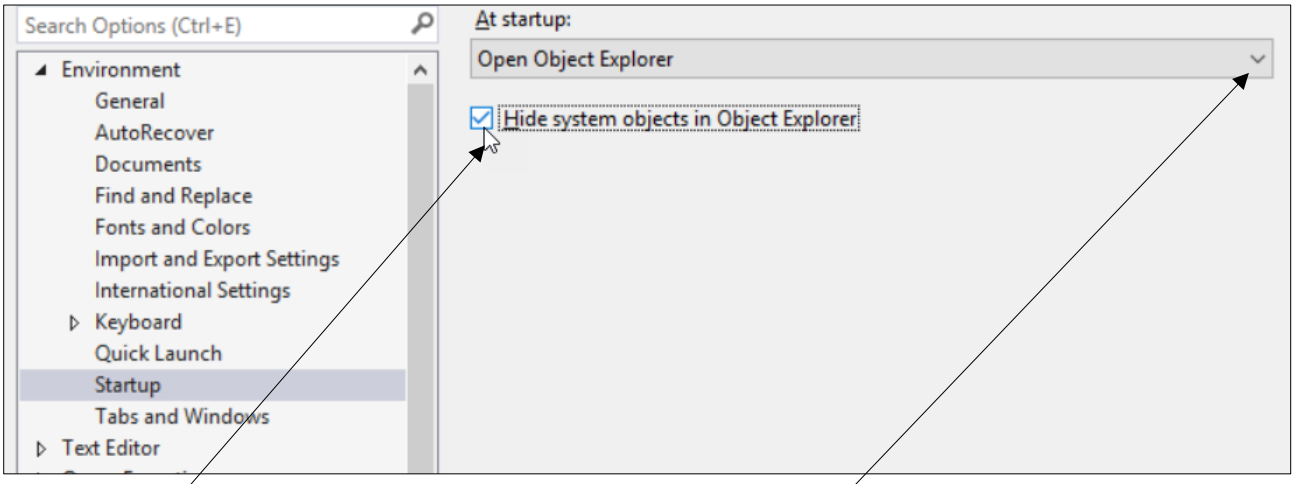
You can click on the ⊞ symbol to expand:

- **Databases**, to see the **Movies** database; then
- **Movies**, to see what it contains; then
- **Tables**, to see what tables there are.



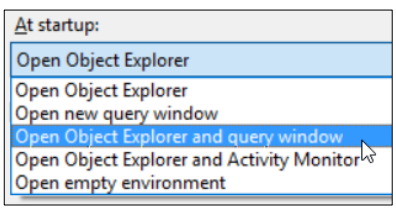
Useful Start-up Options

You can control what happens when you start Management Studio. To do this, from the menu select **Tools ➔ Options**, then complete the dialog box which appears as follows:

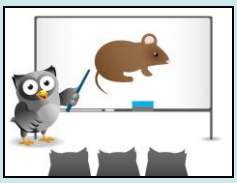


a) System objects clutter up SQL Server, and (in this owl's opinion) are best hidden, although you won't see a huge amount of difference.

b) You can click on the drop arrow and choose (for example) to show a blank query as well as Object Explorer whenever you open Management Studio.



WHAT WE DO

		 ONLINE TRAINING	 MANCHESTER OR LONDON	 AT YOUR OFFICE	 BESPOKE CONSULTANCY
OFFICE 365	Microsoft Excel	✓	✓	✓	✓
	VBA macros	✓	✓	✓	✓
	Office Scripts	✓		✓	
	Microsoft Access				✓
POWER PLATFORM	Power BI and DAX	✓	✓	✓	✓
	Power Apps	✓		✓	
	Power Automate	✓	✓	✓	✓
SQL SERVER	Reporting Services	✓	✓	✓	✓
	Report Builder	✓		✓	✓
	Integration Services	✓	✓	✓	✓
	Analysis Services	✓		✓	
CODING LANGUAGES	SQL	✓	✓	✓	✓
	Visual C#	✓	✓	✓	✓
	Python	✓	✓	✓	✓

Wise Owl Training


Wise Owl
Training