Module 1: Introduction to Microsoft SQL Server

Contents:

Module Overview

This module provides an overview of Microsoft® SQL Server®, the data management software that stores data securely. Before you start, it is helpful to understand the basic architecture of SQL Server, the different editions that are available, and a little about SQL Server Management Studio (SSMS). SSMS is one of the tools you use to connect to instances of SQL Server, write queries, and view data returned by your queries.

Objectives

After completing this module, you will be able to:

- Describe the architecture of SQL Server.
- Describe the different editions of SQL Server.
- Work with SSMS.

Lesson 1: The Basic Architecture of SQL Server

This lesson explains the basic architecture of Microsoft SQL Server, together with some key concepts. You will learn about SQL Server instances, the services, and how databases are structured. This will help you understand how SQL Server works before you start writing queries.

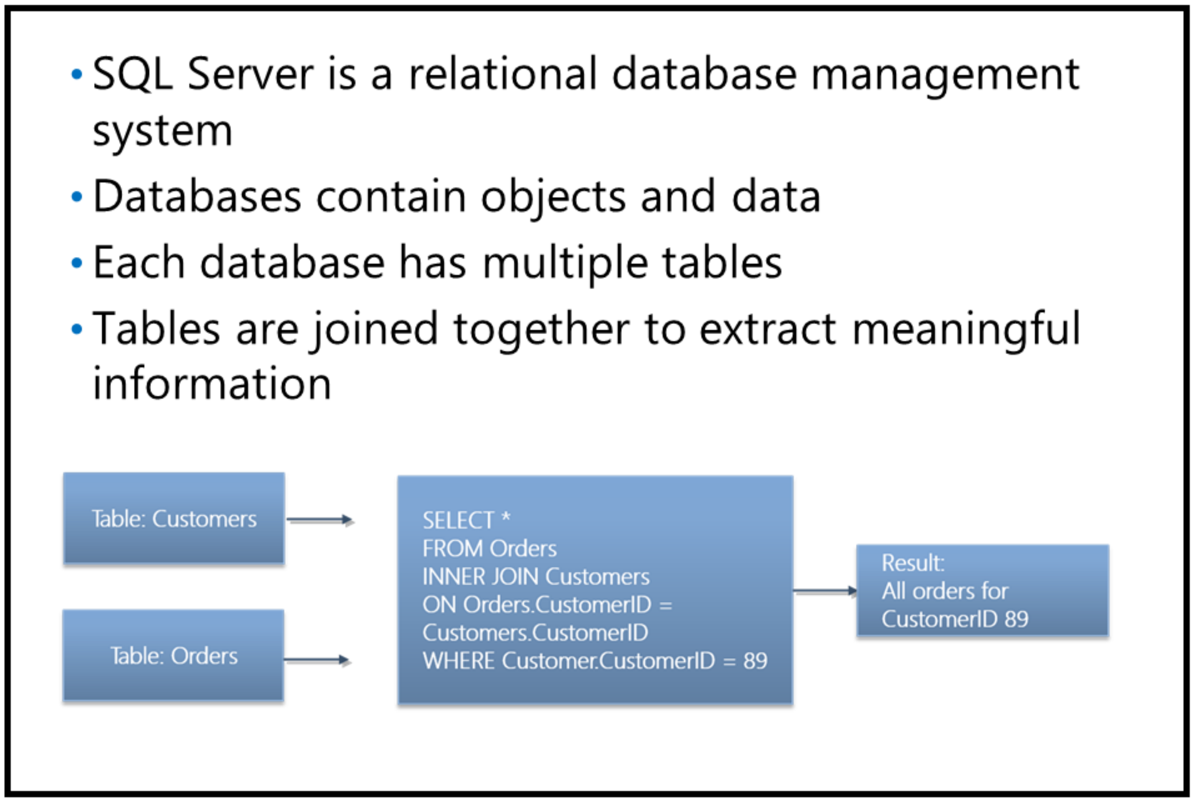
Lesson Objectives

After completing this lesson, you will be able to describe:

- Relational databases in general, and specifically the role and structure of SQL Server databases.
- The sample database used in this course.
- What is meant by the client server model.

- The structure of Transact-SQL (T-SQL) queries.

Relational Databases



- SQL Server is a relational database management system
- Databases contain objects and data
- Each database has multiple tables
- Tables are joined together to extract meaningful information

Table: Customers
Table: Orders
Result: All orders for CustomerID 89

```
SELECT *
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID =
Customers.CustomerID
WHERE Customer.CustomerID = 89
```

SQL Server is a data management system that uses the relational model to store and manage data. Relational databases store information in tables—each table holds information about just one thing. The information may concern something tangible, such as customer details, or intangible things such as orders.

You could hold customer information in the Customers table, but information about the goods they order would be in a separate table called Orders. This way of organizing data is efficient and removes redundant information. However, someone might ask to see all the orders placed by a particular customer. You use SQL Server to get this information by relating these tables to one another. You can then join the two tables together in a query to produce a list of all orders placed by a particular customer.

Databases typically have many different tables related to one another, so you often need to join several tables to obtain the information. For example, you might want to see the orders for customers who buy from one of your salespeople. You can do this by joining the Customers table, the Salesperson table, and the Orders table.

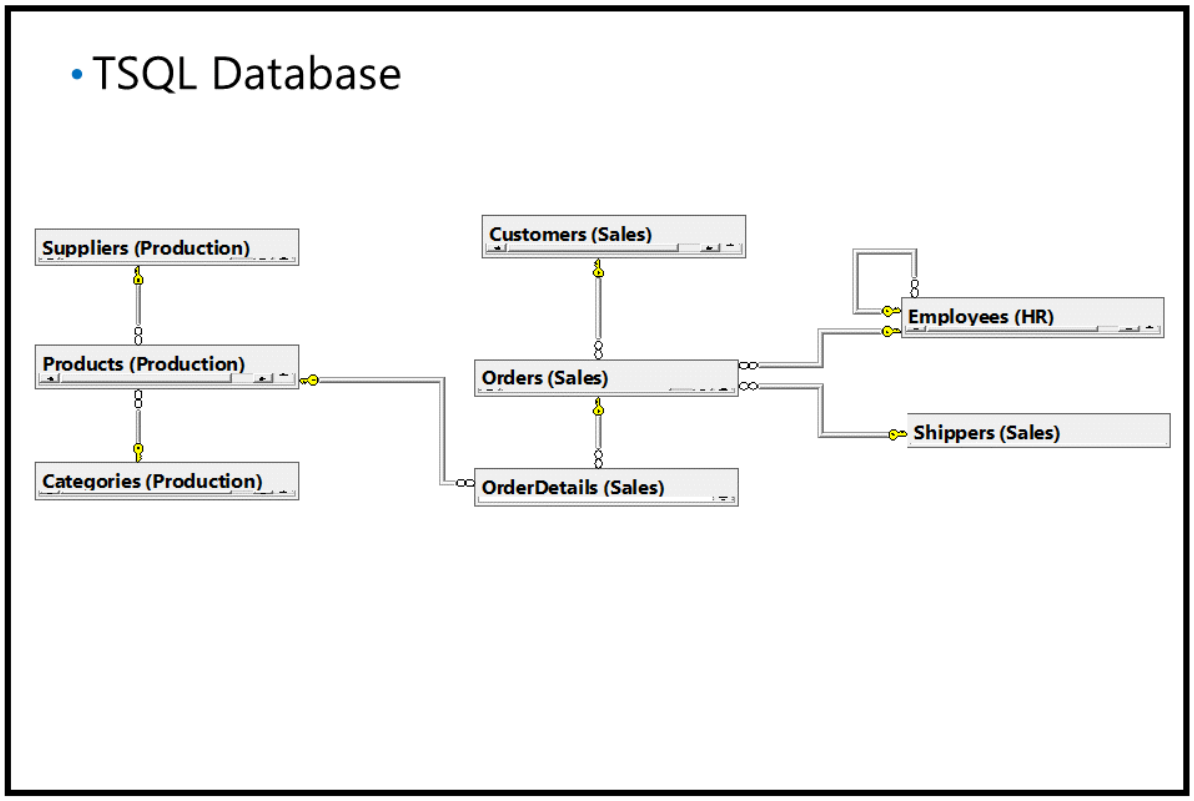
In addition to the databases that are created to store information, SQL Server includes five system databases:

- **master**: the system configuration database.
- **model**: the template database. SQL Server will apply any changes made in model to new databases.
- **msdb**: used by SQL Server Agent to schedule jobs and alerts.
- **tempDb**: a temporary store for data such as work tables. This database is dropped and recreated each time SQL Server restarts, which means that any temporary tables will be lost when SQL Server closes down.

- **resource**: a hidden, read-only database that contains all the system objects for other databases.

SQL Server databases contain data and objects, including tables, views, stored procedures, user accounts, and other management objects. Before you can execute queries, or insert or delete information from a database, you must connect to the database. You need security credentials to log on to SQL Server, and a database account with permissions to access data objects.

About the Course Sample Database



To understand how queries work, you will be using a database called TSQL. This is a small database suitable for learning how to write Transact-SQL queries. TSQL contains several types of objects:

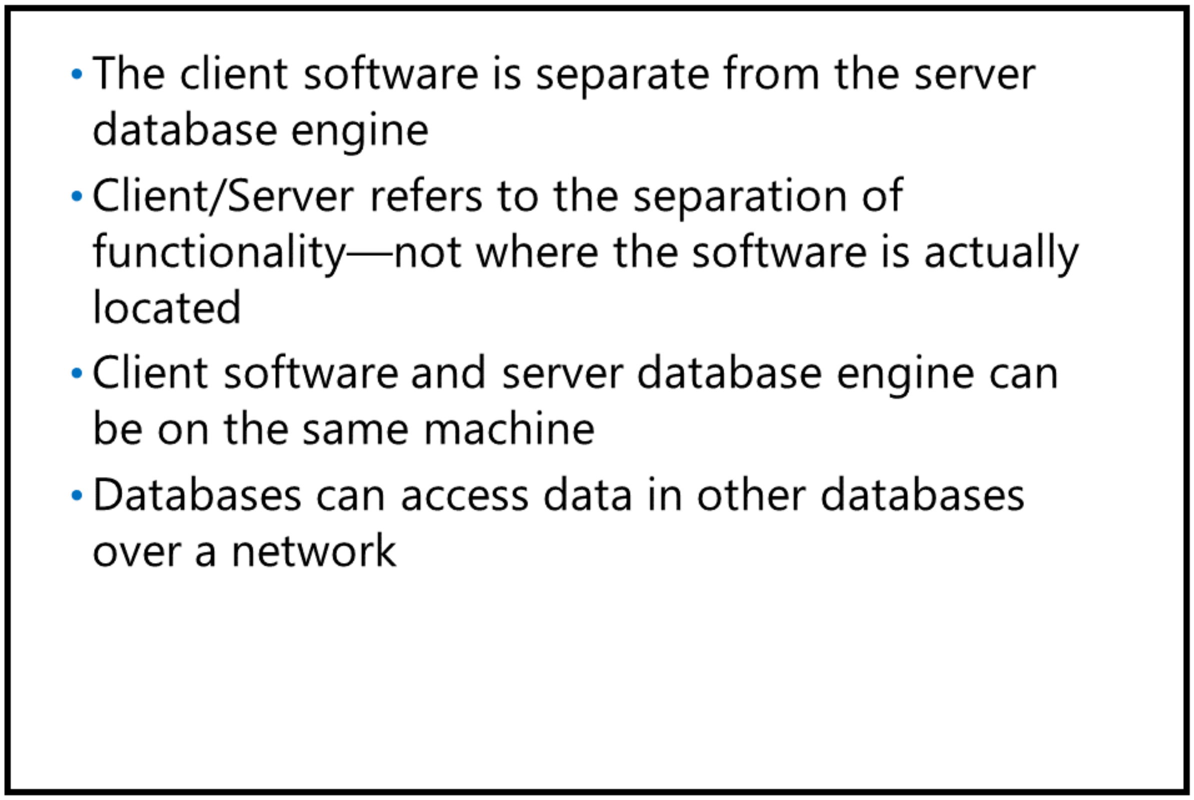
- **Schemas**. These are logical containers for tables and views.
- **Tables**. These mostly relate to one another using Foreign Key constraints.
- **Views**. These display information from more than one table.

The TSQL database is a simple sales application for a small business. Some of the tables you will be working with include:

- **Sales.Orders**. This table stores invoice header information, such as a unique reference for the order, the customer who placed the order, and the date of the order.
- **Sales.OrderDetails**. This table stores transaction details about each order, such as products ordered, and the price.

- **Sales.Customers**. This table stores information about customers, such as company name, and contact details.
- **HR.Employees**. This table stores employee information.

Client Server Databases



- The client software is separate from the server database engine
- Client/Server refers to the separation of functionality—not where the software is actually located
- Client software and server database engine can be on the same machine
- Databases can access data in other databases over a network

SQL Server is a client server system. This means that the client software, which includes SQL Server Management Studio and Visual Studio®, is separate from the SQL Server Database Engine.

When the client application sends requests to the database engine as T-SQL statements, SQL Server performs the necessary file access, memory management, and processor utilization on behalf of the client. The client never has direct access to database files—unlike, for example, a desktop database application.

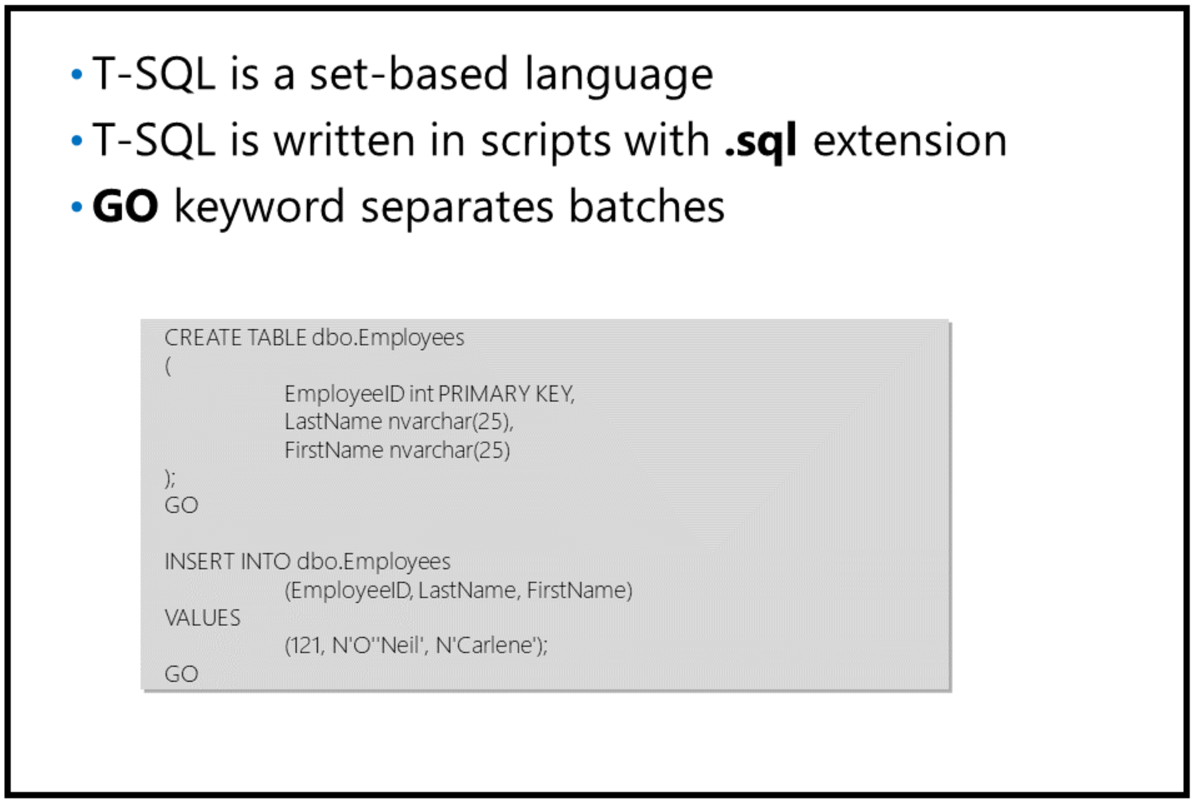
In this course, the client and server software are running on the same virtual machine but, in production environments, the client software runs on a separate machine to the database engine. Indeed, there could be multiple clients accessing the same server database engine.

Wherever the client and server software is located, it makes no difference to the way you write T-SQL code. On the logon screen, you just specify the SQL Server that you want to connect to.

You can also refer to other databases in a T-SQL script by using its four-part name. A four-part name has the format **Instance.Database.Schema.Object**. For example, the four-part name **MIA-SQL.sales.dbo.orders** refers to the orders table, in the dbo schema, in the sales database, on the MIA-SQL server's default instance.

Connecting to a remote server requires the remote instance to be set up as a linked server. In T-SQL, you add a linked server using **sp_addlinkedserver**. Although **sp_addlinkedserver** takes a number of optional arguments, in its simplest form you could connect to the server in the previous example using the statement **exec sp_addlinkedserver in 'MIA-SQL'**.

Queries



- T-SQL is a set-based language
- T-SQL is written in scripts with **.sql** extension
- **GO** keyword separates batches

```
CREATE TABLE dbo.Employees
(
            EmployeeID int PRIMARY KEY,
            LastName nvarchar(25),
            FirstName nvarchar(25)
);
GO

INSERT INTO dbo.Employees
            (EmployeeID, LastName, FirstName)
VALUES
            (121, N'O''Neil', N'Carlene');
GO
```

T-SQL is a set-based language, which means it does not extract data row by row, but instead extracts data from tables that normally contain many rows. Only after it has retrieved the table does SQL Server filter data to produce a subset of the table, if that is what the query has requested. This makes SQL Server highly efficient in dealing with large volumes of data, but it means you have to think in sets to write efficient T-SQL code.

T-SQL scripts are stored in script files with a **.sql** extension. Inside each file, you can divide the script into batches, each batch concluding with the **GO** keyword. SQL Server runs each batch in its entirety before it starts the next one. This is important if you are relying on things happening in a specific order. For example, you must create a table before you can populate the table with data. To complete these two steps within the same script file, you must specify the table structure first, and then add data to the table. If you try to create the table and populate it with data without the **GO** keyword in between, the statement will fail. It will succeed only when the **GO** keyword completes the first **CREATE TABLE** statement before the **INSERT INTO** statement populates the table with data.

Check Your Knowledge

Sequencing Activity

Put the following T-SQL commands in order by numbering each to create a script that will execute without errors:

```
CREATE TABLE HR.Employees
(
EmployeeID int PRIMARY KEY,
LastName nvarchar(25),
FirstName nvarchar(25)
);
```

```
GO
```

```
INSERT INTO HR.Employees
(
EmployeeID, LastName, FirstName
)
VALUES
(121, N'O''Neill, N'Carlene');
```

```
GO
```

Check answer Show solution Reset

Correct

Lesson 2: SQL Server Editions and Versions

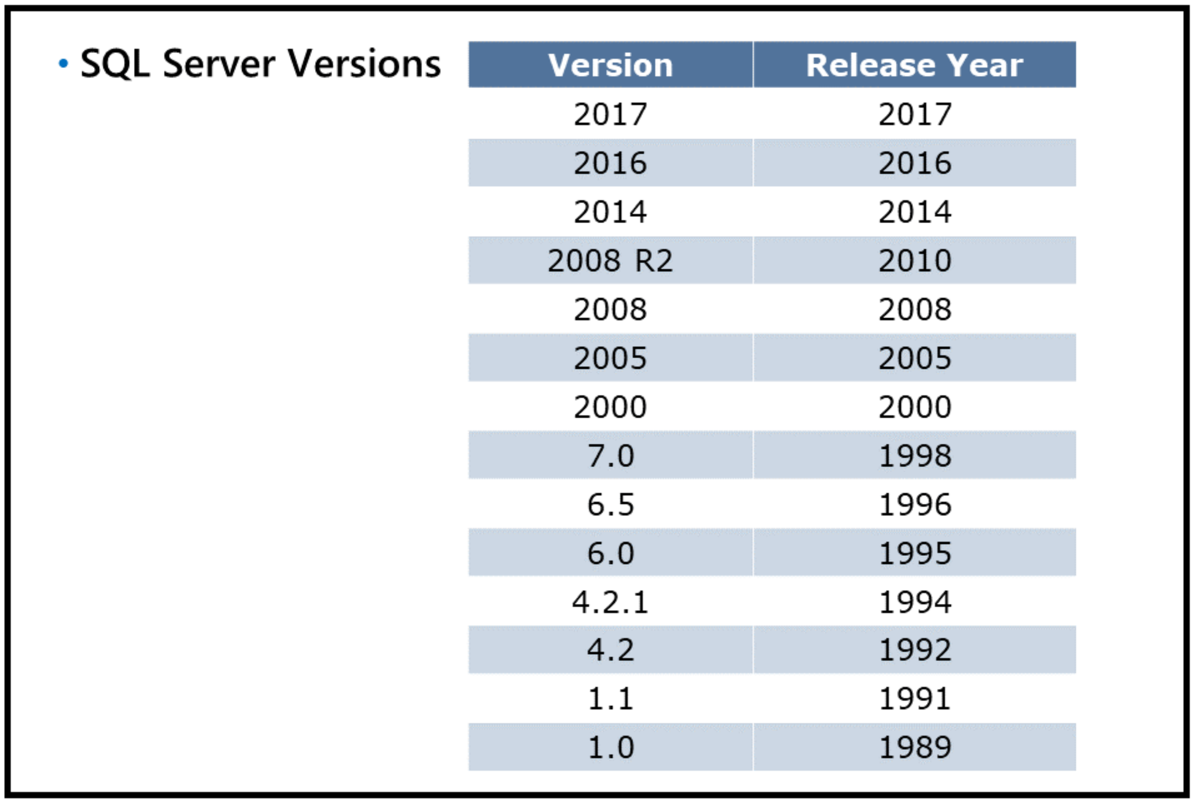
In this lesson, you will learn about the editions and versions of Microsoft SQL Server. You will learn about the different editions, their distinguishing features, and which edition might be best when planning a new deployment.

Lesson Objectives

After completing this lesson, you will be able to describe:

- The versions of SQL Server.
- The editions of SQL Server.

SQL Server Versions



• SQL Server Versions

Version	Release Year
2017	2017
2016	2016
2014	2014
2008 R2	2010
2008	2008
2005	2005
2000	2000
7.0	1998
6.5	1996
6.0	1995
4.2.1	1994
4.2	1992
1.1	1991
1.0	1989

SQL Server 2017 is the latest version in SQL Server's development. Since SQL Server was first developed in 1989 for the OS/2 operating system, it has gone through a number of major releases. SQL Server version 4.2 and later were developed to run on Windows®.

The SQL Server Database Engine had major enhancements for version 7.0 and all subsequent versions have continued to extend and improve SQL Server functionality, making it suitable for workgroup and enterprise use.

SQL Server 2016 was a major new release that included enhanced security, support for hybrid cloud installations, and major improvements in the analytics functionality.

SQL Server 2017 is the first version of SQL Server to run on Linux, and include SQL Graph for enhanced querying and data modelling.

Note: Although the name may suggest otherwise, SQL Server 2008 R2 is not a service pack for SQL Server 2008. SLQ Server 2008 R2 is an independent version (number 10.5) with enhanced multiserver management capabilities, in addition to new business intelligence (BI) features.

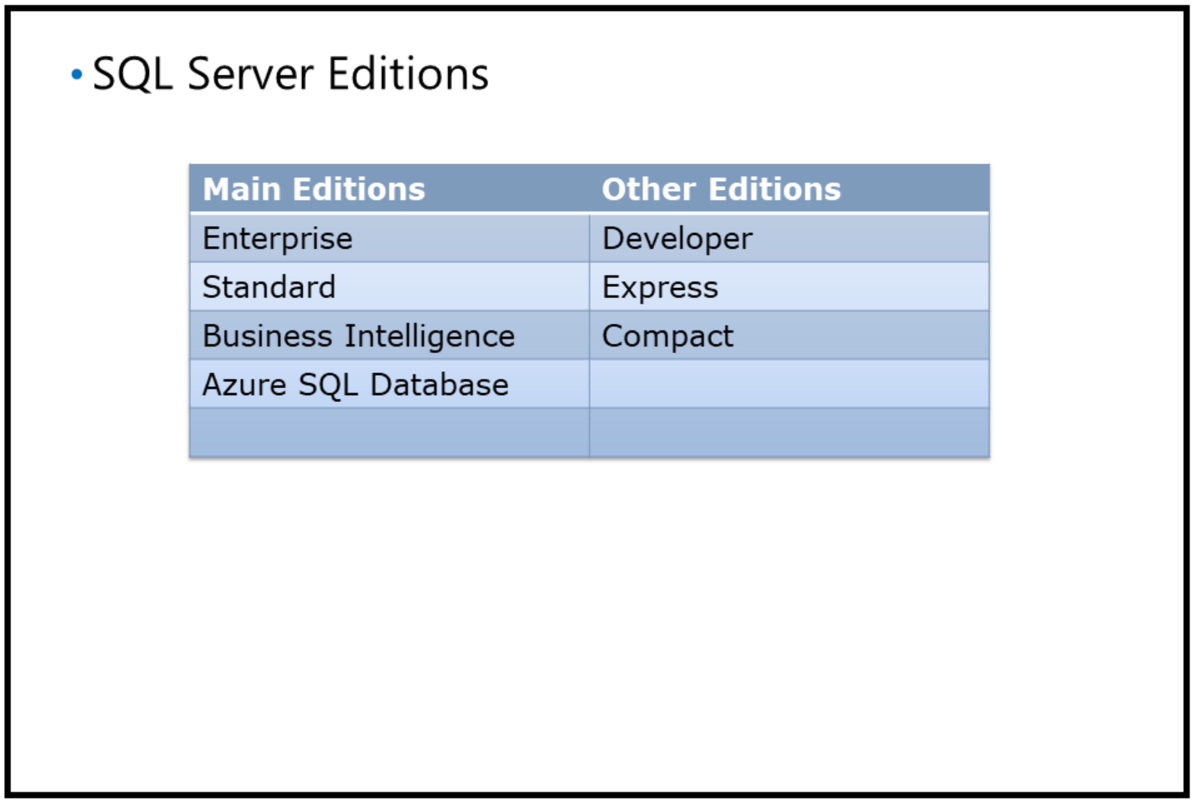
Check Your Knowledge

Discovery

Which version of SQL Server are you currently working with? Have you worked with any earlier versions?

Show solution Reset

SQL Server Editions



• SQL Server Editions

Main Editions	Other Editions
Enterprise	Developer
Standard	Express
Business Intelligence	Compact
Azure SQL Database	

SQL Server is available in different editions with different feature sets that target various business scenarios. In the SQL Server 2012 release, Microsoft streamlined the number of editions from previous versions. Four main editions are available:

- **Enterprise**. This is SQL Server's flagship edition containing all features, including Business Intelligence, support for data warehousing, and high availability.
- **Standard**. This includes the database engine, as well as core reporting and analytics capabilities. Standard supports 16 processor cores but does not include all the high availability, security, and data warehousing features found in the Enterprise edition.
- **Business Intelligence**. This includes the core database engine, along with the full Business Intelligence functionality of analytics, reporting, and integration services. However, like the Standard edition, it supports 16 processor cores and does not offer all the high availability, security, and data warehousing features of the Enterprise edition.
- **Express**. This is a free version of SQL Server and is limited to four processor cores, 1 GB of memory per instance, and 10 GB maximum storage per database.

This course uses features that are found in all editions.

In addition to the editions described above, SQL Server also runs in the cloud, in one of two ways:

- You can install SQL Server on a cloud-based virtual machine that your organization has provisioned and integrated with its infrastructure. When properly set up, SQL Server works as if it were on a server on your network.
- Secondly, it could be an Azure SQL Database. This is Software as a Service (SaaS) and allows you to use SQL

Server without a physical server or a cloud-based virtual machine. You can add or remove performance as required, making this a highly scalable option.

Additional Reading: For more information on the use of Transact-SQL on Microsoft Azure SQL Server Database, see the Microsoft article *Resolving Transact-SQL differences during migration to SQL Database* at: **http://aka.ms/ybpqh8**

For more information about migrating to Azure SQL Server, see the Azure documentation:

Resolving Transact-SQL differences during migration to SQL Database

http://aka.ms/ybpqh8

Check Your Knowledge

Select the best answer

You have founded a new company with two friends. Your new application (app) uses a SQL Server database to store information. You are unsure whether your app will be successful but, if it is, you will need both high performance and space for large volumes of data. However, you have not yet launched, so are unsure how many people will use your app. Which edition of SQL Server should you use for this system?

Azure SQL Database

Enterprise edition

Express edition

Business Intelligence edition

Any edition is appropriate for these requirements

Check answer Show solution Reset

Azure SQL Database allows you to start small, and scale up as required. As a startup, using Azure SQL Database also means you do not need to buy server hardware to run SQL Server. Because Azure SQL Database is Software as a Service (SaaS), you pay for what you use, without high upfront costs.

Lesson 3: Getting Started with SQL Server Management Studio

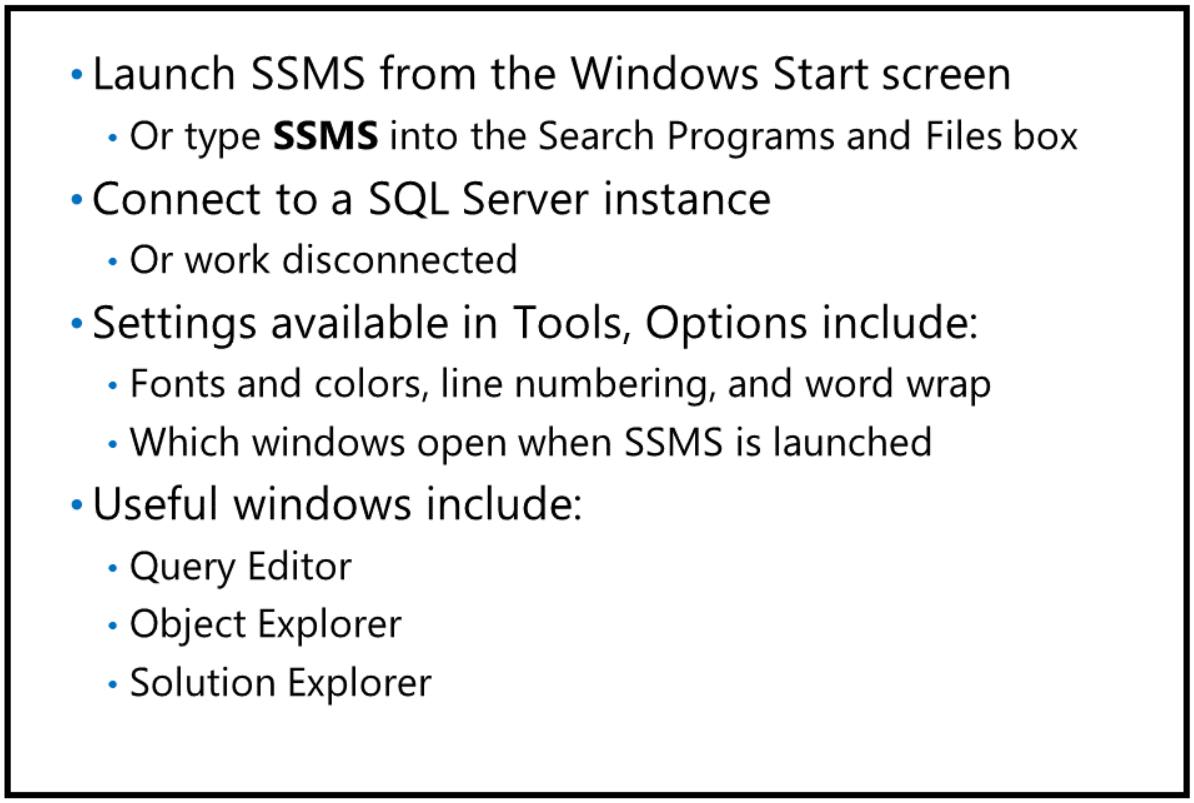
In this lesson, you will learn how to use SQL Server Management Studio (SSMS) to connect to an instance of SQL Server. You will explore the databases contained in the instance and work with script files containing T-SQL queries.

Lesson Objectives

After completing this lesson you will be able to:

- Start SSMS.
- Use SSMS to connect to on-premises SQL Server instances.
- Explore a SQL Server instance using Object Explorer.
- Create and organize script files.
- Execute T-SQL queries.
- Use SQL Server documentation.

Starting SSMS



- Launch SSMS from the Windows Start screen
  - Or type **SSMS** into the Search Programs and Files box
- Connect to a SQL Server instance
  - Or work disconnected
- Settings available in Tools, Options include:
  - Fonts and colors, line numbering, and word wrap
  - Which windows open when SSMS is launched
- Useful windows include:
  - Query Editor
  - Object Explorer
  - Solution Explorer

SSMS is an integrated management, development, and querying application that has many features for exploring and working with your databases. Microsoft based SSMS on the Visual Studio shell; if you know Visual Studio, you will most likely feel comfortable using SSMS.

You can start SSMS in one of two ways:

- Use the **SSMS** shortcut on the Windows Start menu.
- Type **ssms.exe** in a command prompt window.

By default, SSMS will display a Connect to Server dialog box where you can specify the server (or instance) name, together with your security credentials. To specify the database you want to connect to, click the **Options** button to open the Connection properties dialog box. Alternatively, you can select the database after you have connected.

You can explore many SSMS features without connecting to a SQL Server instance. You can also cancel the Connect to Server dialog box if you want to connect to a server later.

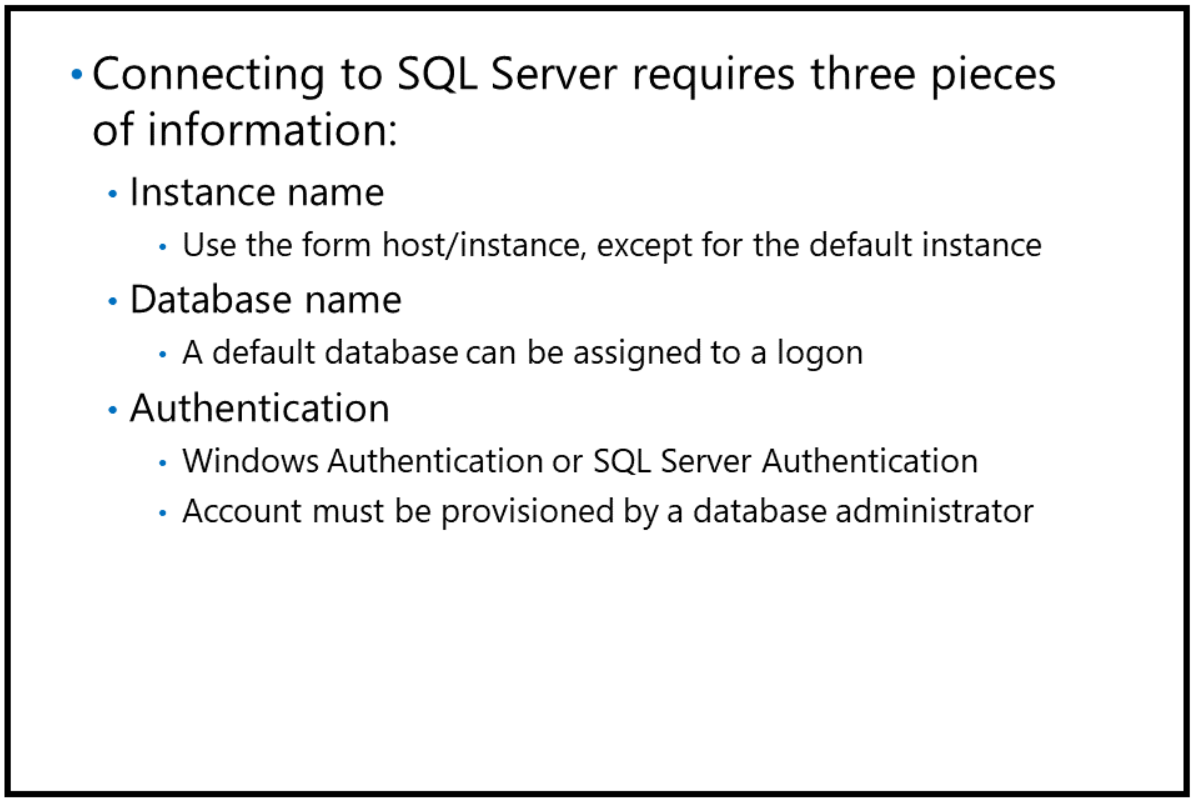
With SSMS running, you can explore some of its settings found in Tools, Options. You can change the default font, enable line numbering for scripts, and control the behavior of its many windows.

For more information on using SSMS, see *Use SQL Server Management Studio* in Microsoft Docs:

Use SQL Server Management Studio

http://aka.ms/cbalxi

Connecting to SQL Server



- Connecting to SQL Server requires three pieces of information:
  - Instance name
    - Use the form host/instance, except for the default instance
  - Database name
    - A default database can be assigned to a logon
  - Authentication
    - Windows Authentication or SQL Server Authentication
    - Account must be provisioned by a database administrator

To connect to an instance of SQL Server, you need to specify several items, regardless of how you connect:

- The instance you want to connect to. This must be in the format: **hostname\instancename**.
- For example, MIA-SQL\Proseware would connect to the Proseware instance on the Windows server named MIA-SQL. If you are connecting to the default instance, you may omit the instance name.
- The name of the database. If you do not specify a database, you will connect to the default database designated by the database administrator. If no default is assigned, you will connect to the master database.

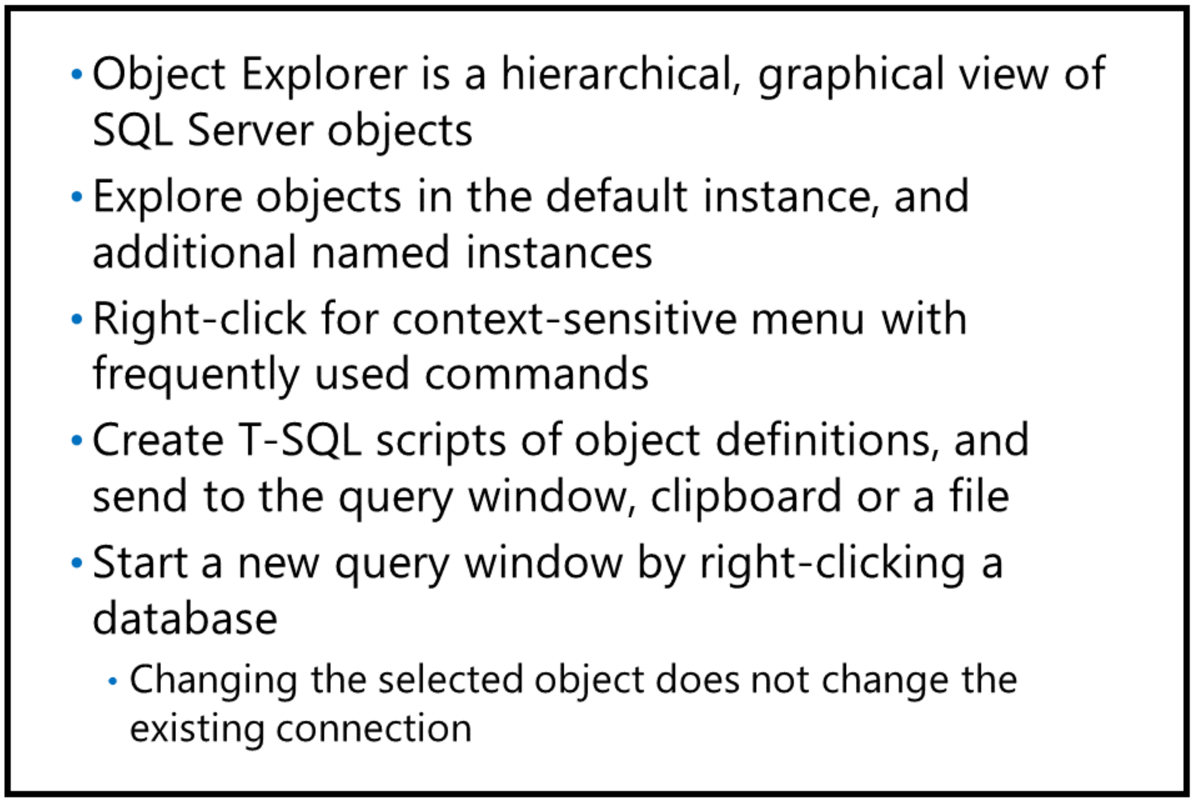
Check Your Knowledge

Discovery

In your organization, which authentication method do you use to log on to SQL Server?

Show solution Reset

Working with Object Explorer



- Object Explorer is a hierarchical, graphical view of SQL Server objects
- Explore objects in the default instance, and additional named instances
- Right-click for context-sensitive menu with frequently used commands
- Create T-SQL scripts of object definitions, and send to the query window, clipboard or a file
- Start a new query window by right-clicking a database
  - Changing the selected object does not change the existing connection

Object Explorer is a graphical tool for managing SQL Server instances and databases. It is one of several SSMS windows available from the View menu. Object Explorer provides direct interaction with most SQL Server data objects such as tables, views, and stored procedures. To display context-sensitive help for an object in the tree view, right-click an object, such as a table. The available options include query and script generators for object definitions.

Note: Operations performed in SSMS require appropriate permissions granted by a database administrator. Being able to see an object or command does not necessarily imply permission to use the object or issue the command.

Use Object Explorer to learn about the structure and definition of data objects you want to use in your queries. For example, to see the names of the columns in a table:

1. Connect to SQL Server, if necessary.
2. Expand the Databases folder to display the list of databases.
3. Expand the relevant database to display the Tables folder.
4. Expand the Tables folder to display the list of tables in the database.
5.

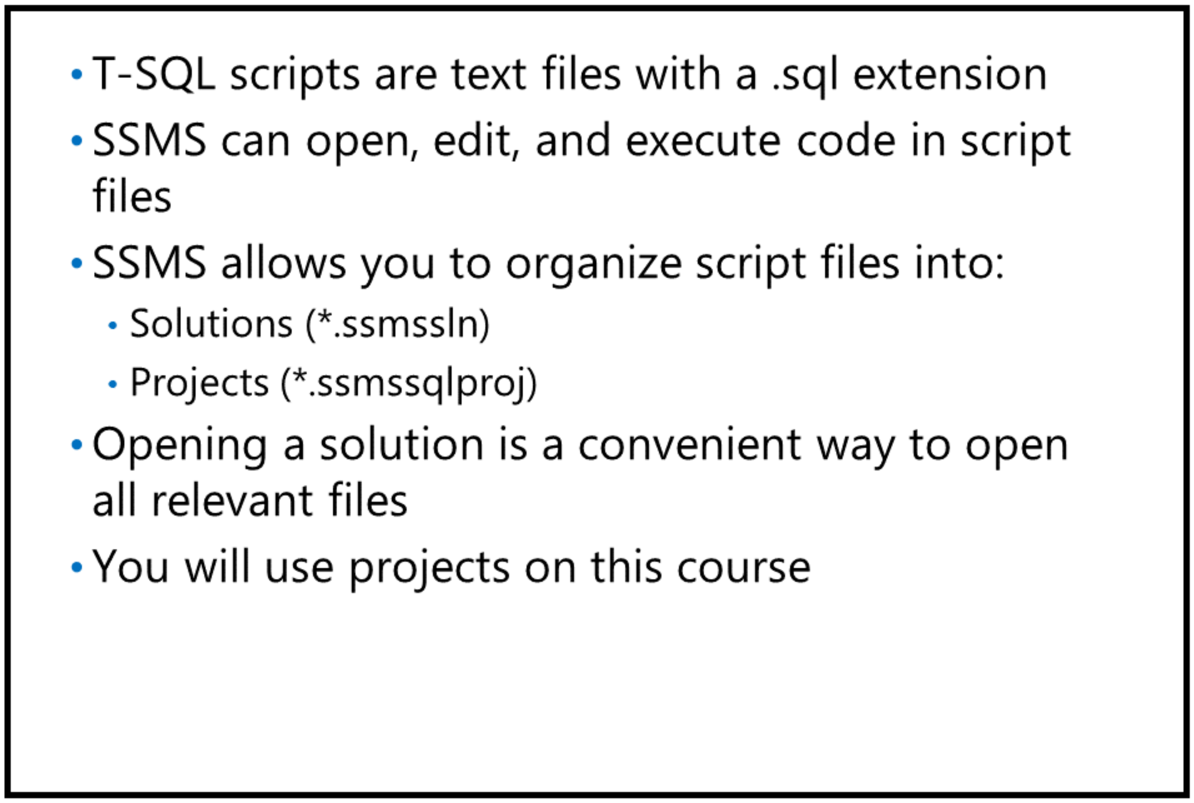
Locate the relevant table and expand it to find the Columns folder.

6. The Columns folder displays the names, data types, constraints, and other information about the column definition.

7. To avoid typing, drag an object from the Object Explorer hierarchy into the query window.

Note: Selecting objects in the Object Explorer pane does not change any connections made in other windows.

Script Files and Projects



- T-SQL scripts are text files with a .sql extension
- SSMS can open, edit, and execute code in script files
- SSMS allows you to organize script files into:
  - Solutions (*.ssmssln)
  - Projects (*.ssmssqlproj)
- Opening a solution is a convenient way to open all relevant files
- You will use projects on this course

SSMS allows you to create and save T-SQL code in text files with a **.sql** file extension. Like other Windows applications that open, edit, and save files, SSMS has both a File menu and toolbar buttons.

In addition to working on individual script files, SSMS lets you organize files into solutions and projects. You can keep scripts for one project together, saving time by opening or closing all the files at the same time. You can open solutions, projects, or script files from SSMS or File Explorer.

Object	Parent	Description
Solution	-	A solution is a conceptual container for projects. Solutions have a **.ssmssln** extension, and are always displayed at the top of the hierarchy.
Project	Solution	Projects contain queries (T-SQL scripts), database connection metadata, and other miscellaneous files. You can file any number of projects within a solution. Projects have a **.ssmssqlproj** extension.
Script	Project	T-SQL script files with a **.sql** extension are the basic files used to work with SQL Server.

To create a new solution, click the **File** menu and click **New Project**. There is no “New Solution” command; if you want a new solution, select **Create New Solution** in the **New Project** dialog box. Type the name for the project, the parent solution, and whether you want the project to be stored in a subfolder within the solution. Click **OK** to create the objects.

You can view solutions and projects by opening the View menu, and selecting Solution Explorer. Solution Explorer displays a hierarchical list of all the solutions and projects you have created.

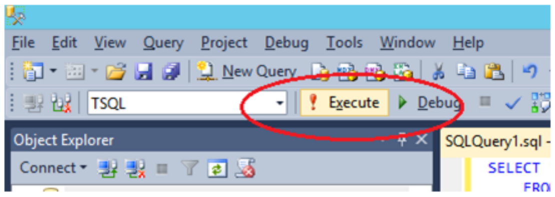
To create a new script that will be stored as part of a project, right-click the **Queries** folder in the **Project** and select **New Query**.

Note: When you create a new query using the toolbar button or the New Query command on the File menu, the script file is stored in the Miscellaneous Files folder by default. Use the **Save As** menu option to save the file in your preferred location. You can drag files from the Miscellaneous Files folder to specific projects, using Solution Explorer to put a copy of the file into a specific project folder. Alternatively, you can move the file by holding the Alt key as you drag.

Remember to save the solution when exiting SSMS, or when opening another solution. When you save a script using the Save toolbar button or the Save <queryname>.sql command on the File menu, you are only saving changes to the current script file. To save the solution and its contents, use the Save All command on the File menu or, when prompted, save the **.ssmssln** and **.ssmssqlproj** files on exit.

Executing Queries

- To execute queries in SSMS:
  - Open a saved script, or write a new query
  - Three ways to execute the query:
    - From the Query menu, select **Execute**
    - Press **F5**
    - Click the **Execute** toolbar button



To execute T-SQL code in SSMS, open the .sql file that contains the query, or type your query into a new query window. You can either run all of the script or part of it:

- Select the portion of code you wish to execute.

- If you do not select anything, the entire script will run.

When you have decided what you wish to execute, run the code by either:

- Clicking the **Execute** button on the SSMS toolbar.
- Clicking the **Query** menu, and then clicking **Execute**.
- Pressing the F5 key, the Alt+X keyboard shortcut, or the Ctrl+E keyboard shortcut.

By default, SSMS will display the results in a new pane of the query window. To change the location and appearance of the results, click **Tools**, and then click **Options**. CTRL-R toggles between a full screen T-SQL editor, and the T-SQL editor plus the results pane.

SSMS enables results to be displayed in three different ways:

- **Grid**. A spreadsheet-like view, with row numbers and columns you can resize. Use **Ctrl+D** to select Grid layout before executing the query.
- **Text**. A Windows Notepad-like display that pads column widths. Use **Ctrl+T** to select text layout before executing the query.
- **File**. Saves query results to a text file with a **.rpt** extension. When you execute the query, you will be prompted for a location to save the file. You can then open the file with any application that reads text files, such as Windows Notepad and SSMS. To send results to file, use the keyboard shortcut **Ctrl-Shift-F** before running the query.

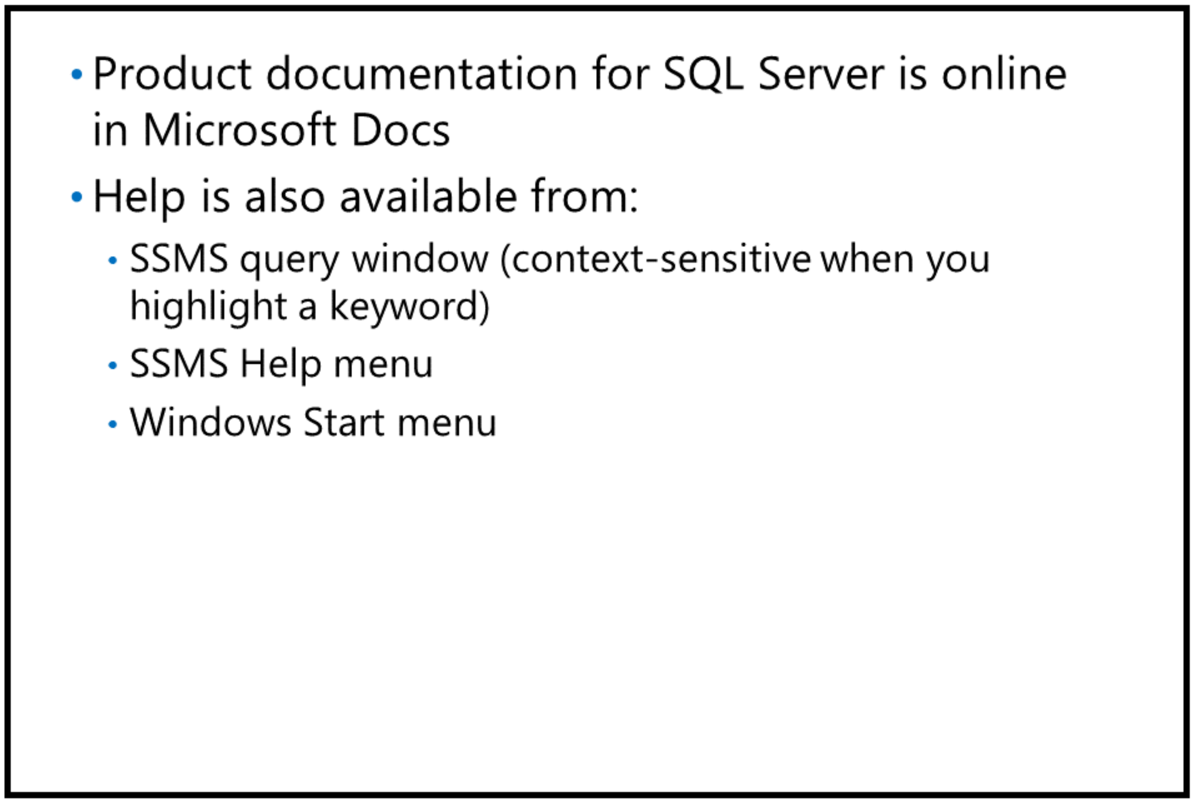
Note: The shortcut to display results as text is **Ctrl+T**. (It had previously been Ctrl+F.)

Additional Reading: For a list of keyboard shortcuts available in SSMS, see *SQL Server Management Studio Keyboard Shortcuts*, in Microsoft Docs:

SQL Server Management Studio Keyboard Shortcuts

http://aka.ms/y83i8i

Using SQL Server Technical Documentation



- Product documentation for SQL Server is online in Microsoft Docs
- Help is also available from:
  - SSMS query window (context-sensitive when you highlight a keyword)
  - SSMS Help menu
  - Windows Start menu

SQL Server Technical Documentation includes help about SQL Server's architecture and concepts, in addition to syntax reference for T-SQL. You can access SQL Server Technical Documentation from the Help menu in SSMS, or from the query window. For context-sensitive help for T-SQL keywords, select the keyword and press F1.

For the latest information, always view SQL Server Technical Documentation in Microsoft Docs:

SQL Server Technical Documentation

http://aka.ms/dxlgjb

Note: SQL Server Technical Documentation is best viewed online. It is constantly being updated, so you know you are accessing the latest information.

For information about installing the Help Viewer and Offline Content for SQL Server, see Microsoft Docs:

Help Viewer and Offline Content for SQL Server

https://aka.ms/Jwdz1n

Before SQL Server 2014, you could install SQL Server Books Online locally:

Get Started with Product Documentation for SQL Server

http://aka.ms/tgv2o6

Demonstration: Introducing Microsoft SQL Server

In this demonstration, you will see how to:

- Use SSMS to connect to an on-premises instance of SQL Server.
- Explore databases and other objects.
- Work with T-SQL scripts.

Demonstration Steps

Use SSMS to Connect to an On-premises instance of SQL Server

1. Ensure that the **MT17B-WS2016-NAT**, **20761C-MIA-DC** and **20761C-MIA-SQL** virtual machines are running.
2. Log on to **20761C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
3. In the **D:\Demofiles\Mod01** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
4. In the **User Account Control** dialog box, click **Yes**, press **y** when prompted, and then press Enter.
5. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.

Explore Database and Other Objects

1. In Object Explorer, expand the **Databases** folder to see a list of databases.
2. Expand the **TSQL** database.
3. Expand the **Tables** folder.
4. Expand the **Sales.Customers** table.
5. Expand the **Columns** folders.
6. View the list of columns, and the data type information for each column.
7. Note the data type for the **companyname** column.

Work with T-SQL Scripts

1. If the Solution Explorer pane is not visible, on the **View** menu, click **Solution Explorer**.
2. In Solution Explorer, notice it will is empty.
3. On the **File** menu, point to **New**, and then click **Project**.
4. In the **New Project** dialog box, under **Installed**, click **SQL Server Management Studio Projects**.
5. In the middle pane, click **SQL Server Scripts**.
6. In the **Name** box, type **Module 1 Demonstration**.

7. In the **Location** box, type **D:\Demofiles\Mod01**.
8. Point out the solution name, and then click **OK**.
9. In Solution Explorer, right-click **Queries**, and click **New Query**.
10. Type the following T-SQL code:

```
USE TSQL;
GO
SELECT CustID, ShipCountry
FROM Sales.Orders;
```

11. Select the code and click **Execute**.
12. Point out the **Results** pane.
13. On the **File** menu, click **Save All**.
14. On the **File** menu, click **Close Solution**.
15. On the **File** menu, point to **Recent Projects and Solutions**, and then click **1 D:\...\ Module 1 Demonstration\Module 1 Demonstration.ssmssln**.
16. Point out the **Solution Explorer** pane.
17. Close SQL Server Management Studio without saving any files.

Check Your Knowledge

Select the best answer

A colleague has asked you to run some test queries against the company's scheduling database. Administrators have given you the name of the server where the database is hosted, and the name of the database. Permissions to run the necessary queries have been granted to your Active Directory® account. You are logged on to a client computer with this Active Directory account and have started SQL Server Management Studio. What other information do you need to connect to the database?

Your Active Directory account username.

Your Active Directory account password.

The name of the login created for you in the SQL Server instance.

The name of the instance that hosts the database.

The name of the user created for you in the SQL Server database.

Check answer Show solution Reset

The name of the instance that hosts the database.

You do not need to provide your Active Directory credentials because these will be sent to SQL Server automatically when you connect. This account has been associated with a login and user in SQL Server by database administrators when they granted access to your account. However, unless the database is on the default instance, you must specify which instance to connect to.

Lab: Working with SQL Server Tools

Scenario

The Adventure Works Cycles Bicycle Manufacturing Company has adopted SQL Server as its relational database management system. You need to retrieve business data from several SQL Server databases. In the lab, you will begin to explore the new environment, and become acquainted with the tools for querying SQL Server.

Objectives

After completing this lab, you will be able to:

- Use SQL Server Management Studio.
- Create and organize T-SQL scripts.

Lab Setup

Estimated Time: 30 minutes

Virtual machine: **20761C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Working with SQL Server Management Studio

Scenario

You have been tasked with writing queries for SQL Server. Initially, you want to become familiar with the development environment. You have therefore decided to explore SQL Server Management Studio and configure the editor for your use.

The main tasks for this exercise are as follows:

1. Open Microsoft SQL Server Management Studio
2. Configure the Editor Settings

Detailed Steps ▲

Task 1: Open Microsoft SQL Server Management Studio

1. Ensure that the **MT17B-WS2016-NAT**, **20761C-MIA-DC** and **20761C-MIA-SQL** virtual machines are running, and then log on to **20761C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Start SSMS but do not connect to an instance of SQL Server.
3. Close the Object Explorer and Solution Explorer windows.
4. Using the **View** menu, show the Object Explorer and Solution Explorer windows in SSMS.

Detailed Steps ▲

Task 2: Configure the Editor Settings

1. With SSMS running, open the **Tools** menu and choose **Options**. Change the text editor font size to **14** points.
2. Change additional settings in **Options**:
   - Disable IntelliSense.
   - Change the tab indent to 6 spaces.
   - Include column headers when copying the result set from the grid. Select **Query Results**, **SQL Server**, **Results to Grid**. Select **Include column headers when copying or saving the results**.

Result: After this exercise, you should have opened SSMS and configured editor settings.

Exercise 2: Creating and Organizing T-SQL Scripts

Scenario

You have decided to organize your T-SQL scripts in a project folder. In this lab, you will practice how to create a project and add query files to it.

The main tasks for this exercise are as follows:

1. Create a Project
2. Add an Additional Query File
3. Reopen the Created Project

Detailed Steps ▲

Task 1: Create a Project

1. Create a new project called **MyFirstProject** and save it in the folder **D:\Labfiles\Lab01\Starter**.
2. Add a new query called **MyFirstQueryFile.sql** to **MyFirstProject**.
3. Save the project and the query file by clicking **Save All**.

Detailed Steps ▲

Task 2: Add an Additional Query File

1. Add an additional query file called **MySecondQueryFile.sql** to the project you created.
2. Open File Explorer, navigate to the **MyFirstProject** folder to check that your second query file is in your project folder.
3. In SSMS, use the Solution Explorer pane to remove **MySecondQueryFile.sql** from your project by choosing the **Remove** option. (Not the Delete option.)
4. In File Explorer, check to see whether the second query file is still in the project folder.
5. In SSMS, remove **MyFirstQueryFile.sql** by choosing **Delete**.
6. To see the difference, check in File Explorer.

Detailed Steps ▲

Task 3: Reopen the Created Project

1. Save the project, close SSMS, reopen SSMS, and open the project **MyFirstProject**.
2. Drag **MySecondQueryFile.sql** from File Explorer to the Queries folder beneath **MyFirstProject** in SSMS Solution Explorer.
3. Save the project.

Result: After this lab exercise, you will have a basic understanding of how to create a project in SSMS and add T-SQL query files to it.

Module Review and Takeaways

In this module, you have learned how to:

- Describe the architecture of SQL Server.
- Describe the different editions of SQL Server.
- Work with SSMS.

Review Question(s)

Check Your Knowledge

Discovery

Can a SQL Server database be stored across multiple instances?

Show solution Reset

No. A database is completely contained within a single instance.

Check Your Knowledge

Discovery

If no T-SQL code is selected in a query window, which code lines will be run when you click the Execute button?

Show solution Reset

All statements in the script will be executed.

Check Your Knowledge

Discovery

What does a SQL Server Management Studio solution contain?

Show solution Reset

SSMS allows you to organize SQL Scripts so that you can manage large collections of files. Projects can contain scripts, connection strings and other settings. Solutions are collections of projects.