Developing with an Azure SQL Database

Lab Time: 60 minutes

Lab Folder: C:\Student\Modules\04_AzureStorage\Lab

Lab Overview: In this module, you will execute a PowerShell script to create an Azure SQL database named **ProductsDB**. After that, you will extend an ASP.NET MVC application named **ProductManagerSQL** with an Entity Framework model and a strongly-typed controller class to read and write product data in the **ProductsDB** database.

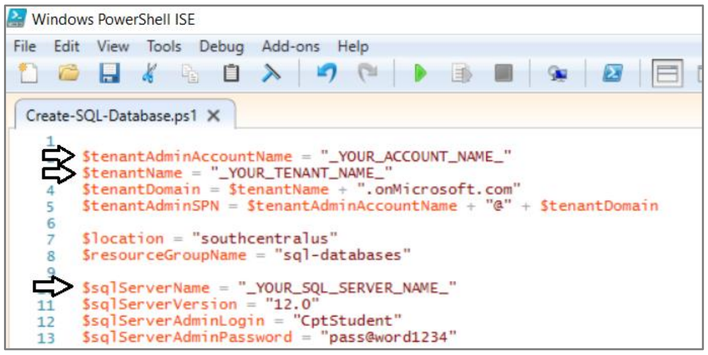
Exercise 1: Deploying an Azure SQL Database

In this exercise you will run a pre-provided PowerShell script to create a new Azure SQL Server instance and an Azure SQL database.

1. Open the PowerShell script named **Create-SQL-Database.ps1**.
   a) Using Windows Explorer locate the PowerShell script named **Create-SQL-Database.ps1** at the following path.

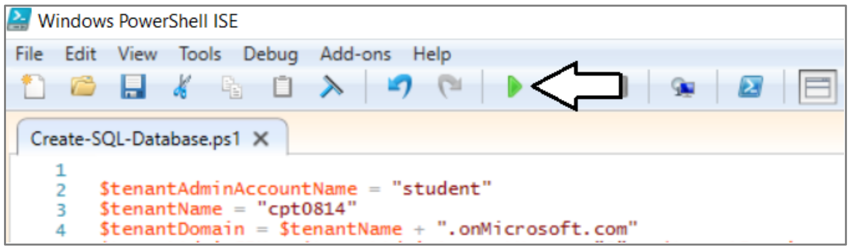
```
C:\Student\Modules\04_AzureStorage\Lab\Scripts\Create-SQL-Database.ps1
```

   b) Right click **Create-SQL-Database.ps1** and then select the **Edit** menu command top open the script in the PowerShell IDE.
2. Take a moment to review the PowerShell code inside this script. You should be able to see it automates the following tasks.
   a) Login to Azure Resource Manager
   b) Create a resource group named **sql-databases** if that resource group doesn't already exist.
   c) Create a SQL Server instance using a new unique name if it doesn't already exist.
   d) Configure access to the SQL Server instance by adding firewall rules.
   e) Create a new SQL database named **ProductsDB** if it doesn't already exist.
3. Modify PowerShell script variables for your environment.
   a) Modify the variable named **$tenantAdminAccountName** with your Azure account name
   b) Modify the variable named **$tenantName** with the name of your tenant.
   c) Modify the variable named **$sqlServerName** with a unique name such as the name of your tenant.



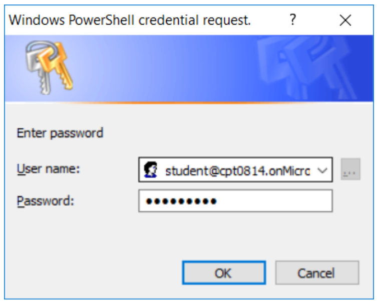
```
$tenantAdminAccountName = "_YOUR_ACCOUNT_NAME_"
$tenantName = "_YOUR_TENANT_NAME_"
$tenantDomain = $tenantName + ".onMicrosoft.com"
$tenantAdminSPN = $tenantAdminAccountName + "@" + $tenantDomain

$location = "southcentralus"
$resourceGroupName = "sql-databases"

$sqlServerName = "_YOUR_SQL_SERVER_NAME_"
$sqlServerVersion = "12.0"
$sqlServerAdminLogin = "CptStudent"
$sqlServerAdminPassword = "pass@word1234"
```

   d) Save your changes to **Create-SQL-Database.ps1**.
4. Execute the PowerShell script named Create-SQL-Database.ps1 to create a new SQL Server and SQL database.
   a) Execute **Create-SQL-Database.ps1** by clicking the **Execute** button with the green arrow button in the PowerShell ISE.



```

$tenantAdminAccountName = "student"
$tenantName = "cpt0814"
$tenantDomain = $tenantName + ".onMicrosoft.com"
```

b) When prompted to log in, enter the user name and password for the account you are using to access your Azure subscription.

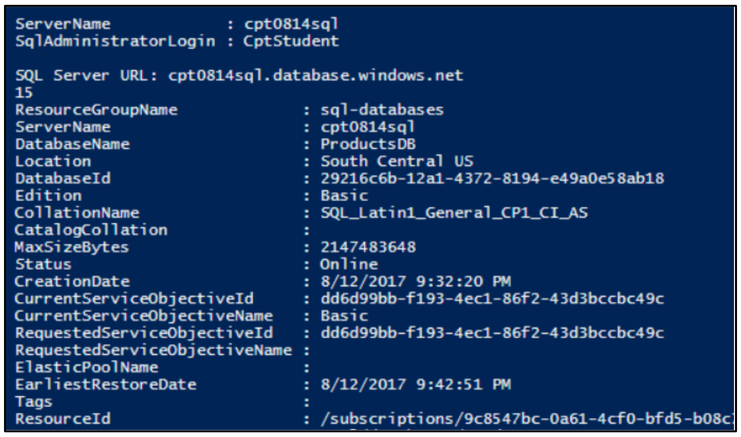

Windows PowerShell credential request.

Enter password

User name: student@cpt0814.onMicro

Password: •••••••••

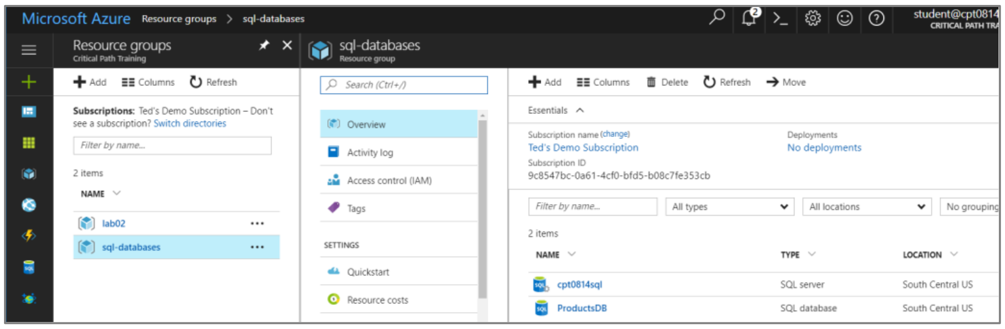
c) The script should execute without error and display output in the console similar to what is shown in the following screenshot.


```
ServerName              : cpt0814sql
SqlAdministratorLogin : CptStudent

SQL Server URL: cpt0814sql.database.windows.net
15
ResourceGroupName                : sql-databases
ServerName                       : cpt0814sql
DatabaseName                     : ProductsDB
Location                         : South Central US
DatabaseId                       : 29216c6b-12a1-4372-8194-e49a0e58ab18
Edition                          : Basic
CollationName                    : SQL_Latin1_General_CP1_CI_AS
CatalogCollation                 :
MaxSizeBytes                     : 2147483648
Status                           : Online
CreationDate                     : 8/12/2017 9:32:20 PM
CurrentServiceObjectiveId        : dd6d99bb-f193-4ec1-86f2-43d3bccbc49c
CurrentServiceObjectiveName      : Basic
RequestedServiceObjectiveId      : dd6d99bb-f193-4ec1-86f2-43d3bccbc49c
RequestedServiceObjectiveName    :
ElasticPoolName                  :
EarliestRestoreDate              : 8/12/2017 9:42:51 PM
Tags                             :
ResourceId                       : /subscriptions/9c8547bc-0a61-4cf0-bfd5-b08c
```

5. Examine what has been created in the Azure portal.

   a) Navigate to the Azure portal at https://portal.azure.com and login with your Azure account credentials.

   b) Locate the resource group named **sql-database**.



Subscriptions: Ted's Demo Subscription – Don't see a subscription? Switch directories

2 items

NAME
lab02
sql-databases

Subscription name (change): Ted's Demo Subscription

Subscription ID: 9c8547bc-0a61-4cf0-bfd5-b08c7fe353cb

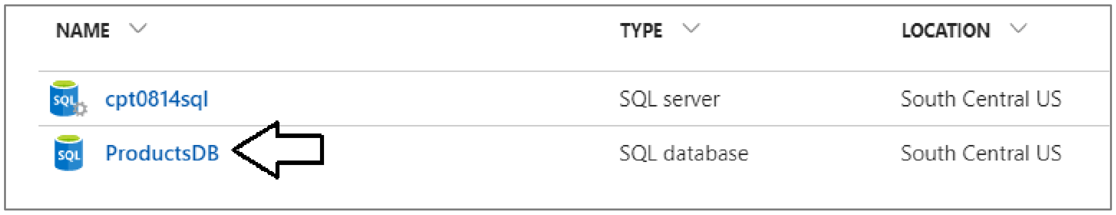
Deployments: No deployments

2 items

NAME	TYPE	LOCATION
cpt0814sql	SQL server	South Central US
ProductsDB	SQL database	South Central US

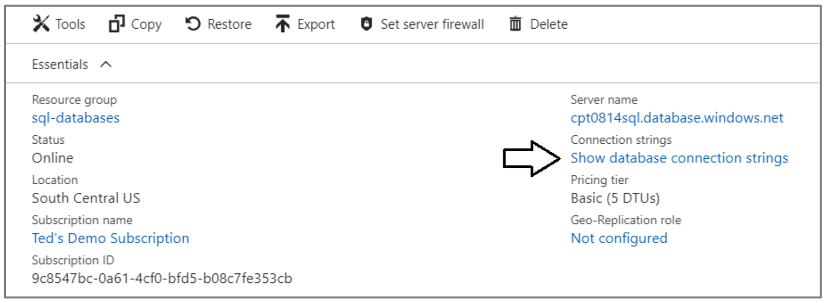
   c) Verify you can see a SQL Server instance and a SQL database named **ProductsDB**.

   d) Click on **ProductsDB** to navigate to the overview page for that SQL database.



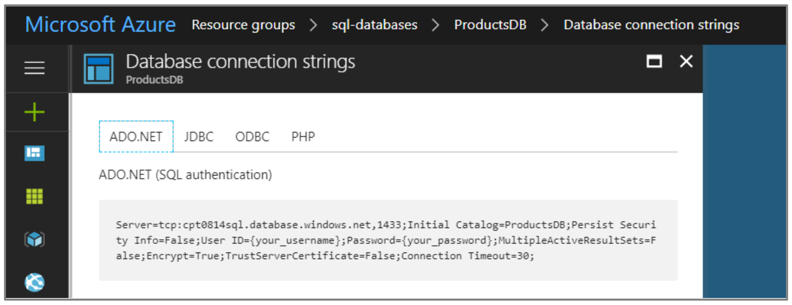
NAME	TYPE	LOCATION
cpt0814sql	SQL server	South Central US
ProductsDB	SQL database	South Central US

e) In the **Essentials** section, click **Show database connection strings**.



Resource group: sql-databases
Status: Online
Location: South Central US
Subscription name: Ted's Demo Subscription
Subscription ID: 9c8547bc-0a61-4cf0-bfd5-b08c7fe353cb
Server name: cpt0814sql.database.windows.net
Connection strings: Show database connection strings
Pricing tier: Basic (5 DTUs)
Geo-Replication role: Not configured

f) Note that you can see the connection string and copy-and-paste it if you need it.

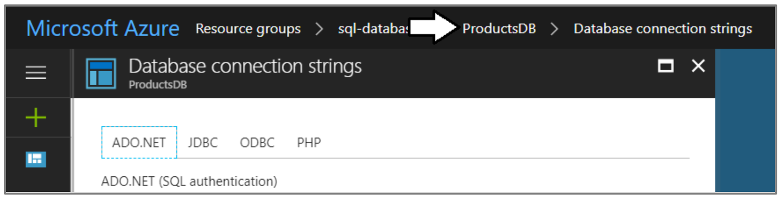

Microsoft Azure
Resource groups > sql-databases > ProductsDB > Database connection strings
Database connection strings
ProductsDB
ADO.NET
JDBC
ODBC
PHP

ADO.NET (SQL authentication)

```
Server=tcp:cpt0814sql.database.windows.net,1433;Initial Catalog=ProductsDB;Persist Securi
ty Info=False;User ID={your_username};Password={your_password};MultipleActiveResultSets=F
alse;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;
```

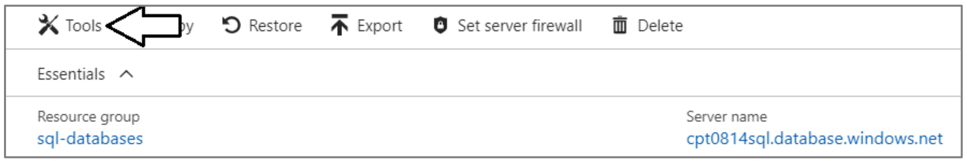
There is no need in this particular lab to copy-and-paste the connection string to this database. However, you should know that you can always use this page to copy the connection string if you ever need it.

g) On the breadcrumb navigation menu at the top of the page, click **ProductsDB** to move back to the database **Overview** page.

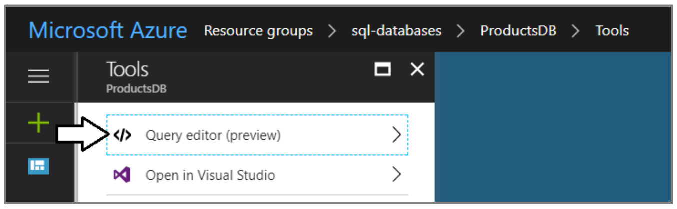

Microsoft Azure
Resource groups > sql-databa ProductsDB > Database connection strings
Database connection strings
ProductsDB
ADO.NET
JDBC
ODBC
PHP
ADO.NET (SQL authentication)

6. Run a query against the database using the new Azure portal Query editor.

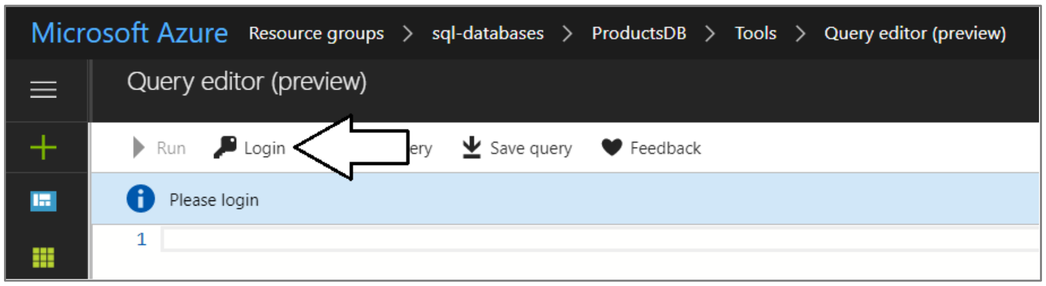
a) On the toolbar of Overview page, click the **Tools** button.



b) Click Query editor to open a new blade with the Query editor.

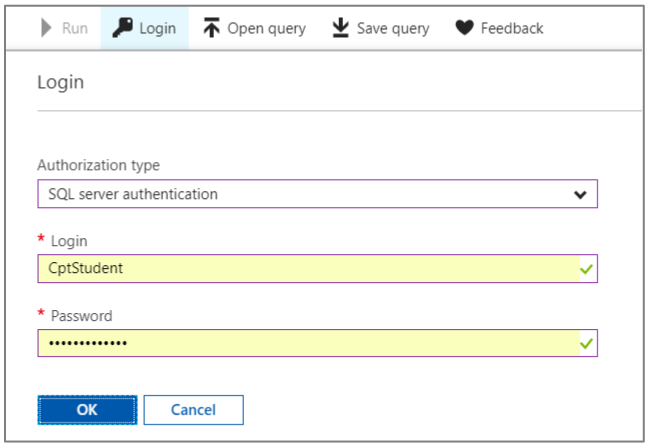


c) Click the Login button to log into the query editor.



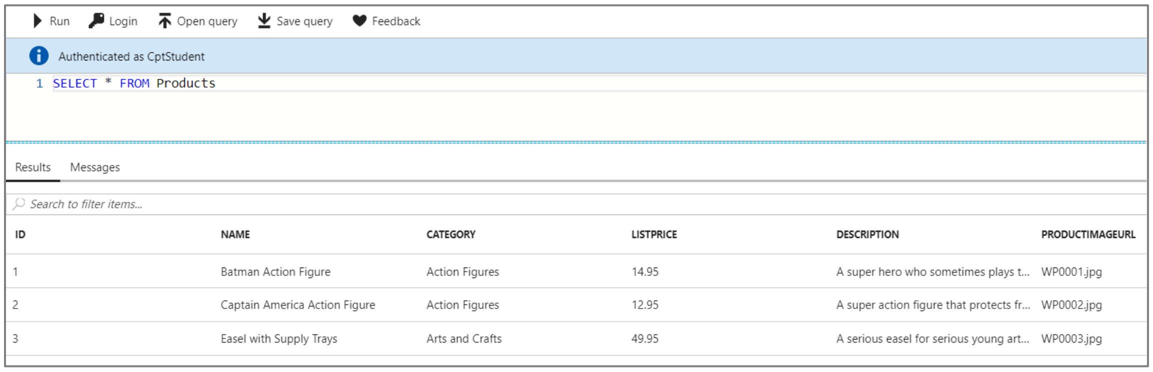
You will not log into the query editor using your Azure account. Instead, you will log in with the SQL admin account that was created by the PowerShell script. The script created the SQL admin with an account name of **CptStudent** and password of **pass@word1234**.

d) Enter a Login of **CptStudent** and Password of **pass@word1234** and then click OK.



Authorization type: SQL server authentication
* Login: CptStudent
* Password: •••••••••••••

e) Enter a simple query of **SELECT * FROM Products** and then click the **Run** button.

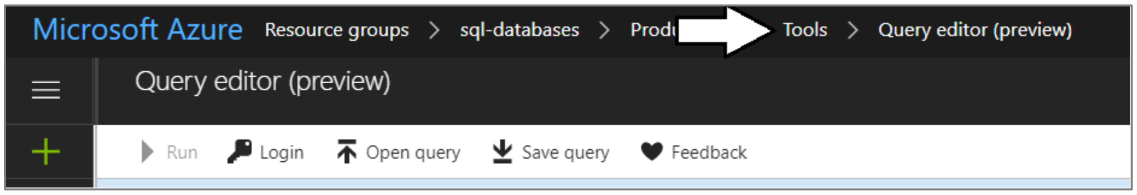


Authenticated as CptStudent

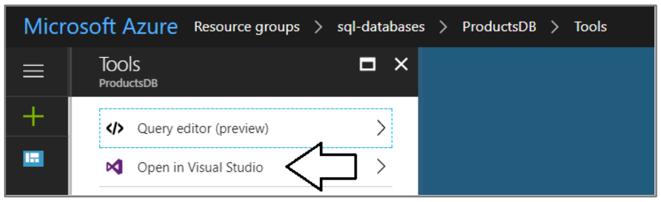
```
SELECT * FROM Products
```

ID	NAME	CATEGORY	LISTPRICE	DESCRIPTION	PRODUCTIMAGEURL
1	Batman Action Figure	Action Figures	14.95	A super hero who sometimes plays t...	WP0001.jpg
2	Captain America Action Figure	Action Figures	12.95	A super action figure that protects fr...	WP0002.jpg
3	Easel with Supply Trays	Arts and Crafts	49.95	A serious easel for serious young art...	WP0003.jpg

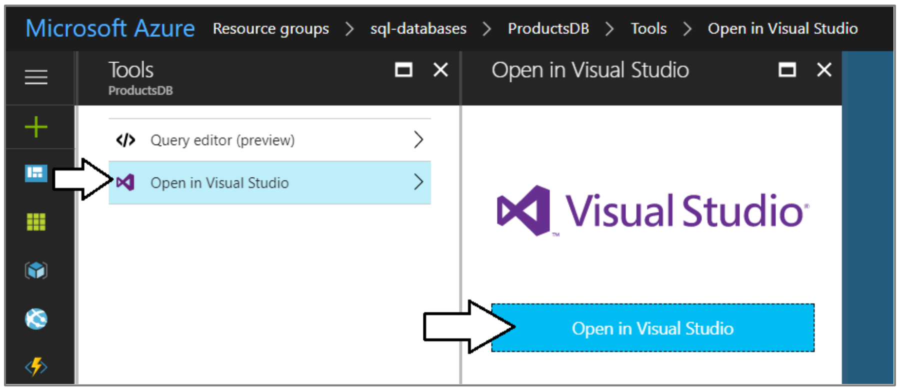
f) After testing the Query editor, click on the **Tools** link in the breadcrumb navigation to navigate back to the **Tools** blade.



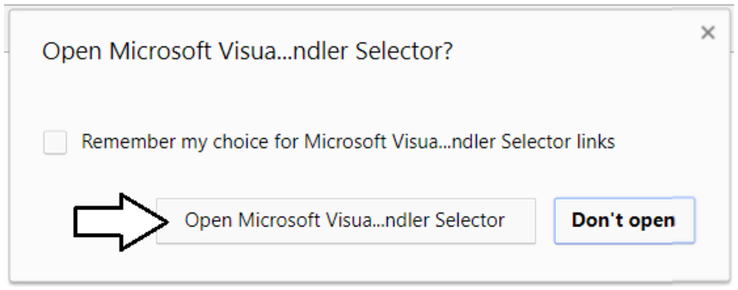
7. Open the **ProductsDB** database in Visual Studio
   a) From the **Tools** blade, click **Open in Visual Studio**.



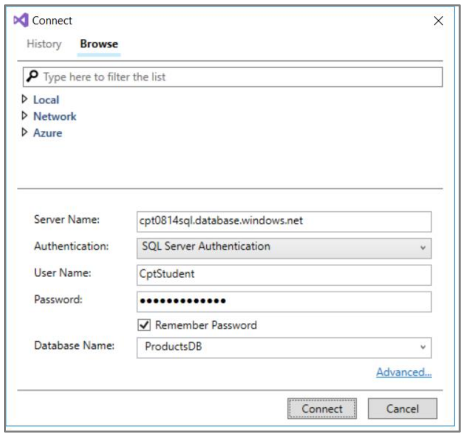
   b) Click the second button that also has the caption **Open in Visual Studio**.



   c) You will be prompted with a dialog box that looks like the following screenshot. Click the **Open** button on the left.


Open Microsoft Visua...ndler Selector?
Remember my choice for Microsoft Visua...ndler Selector links
Open Microsoft Visua...ndler Selector
Don't open

   d) When you are prompted with the Connect dialog, enter a User Name of **CptStudent** and password of **pass@word1234**.

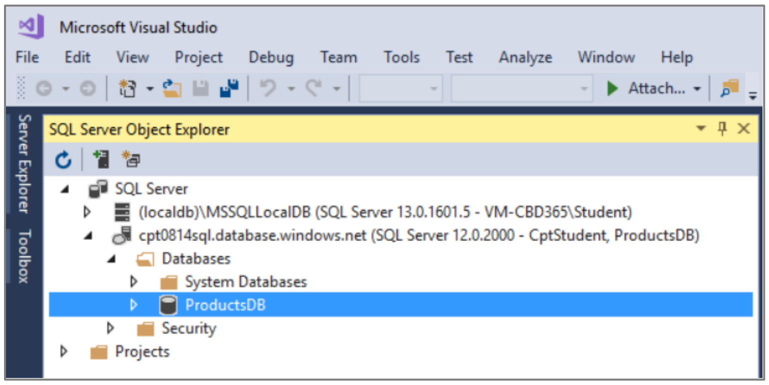


Server Name: cpt0814sql.database.windows.net
Authentication: SQL Server Authentication
User Name: CptStudent
Password: ••••••••••••••
☑ Remember Password
Database Name: ProductsDB

   e) Click to **Connect** button to connect Visual Studio to your new Azure SQL database.

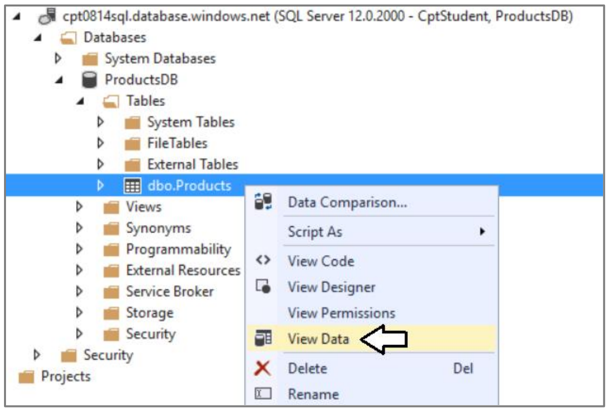
Be patient as it might take Visual Studio 1-2 minutes when it connects to the SQL Azure database.

f) Once Visual Studio has connected to the **ProductsDB** database, it should appear in the **SQL Server Object Explorer** as shown in the following screenshot.

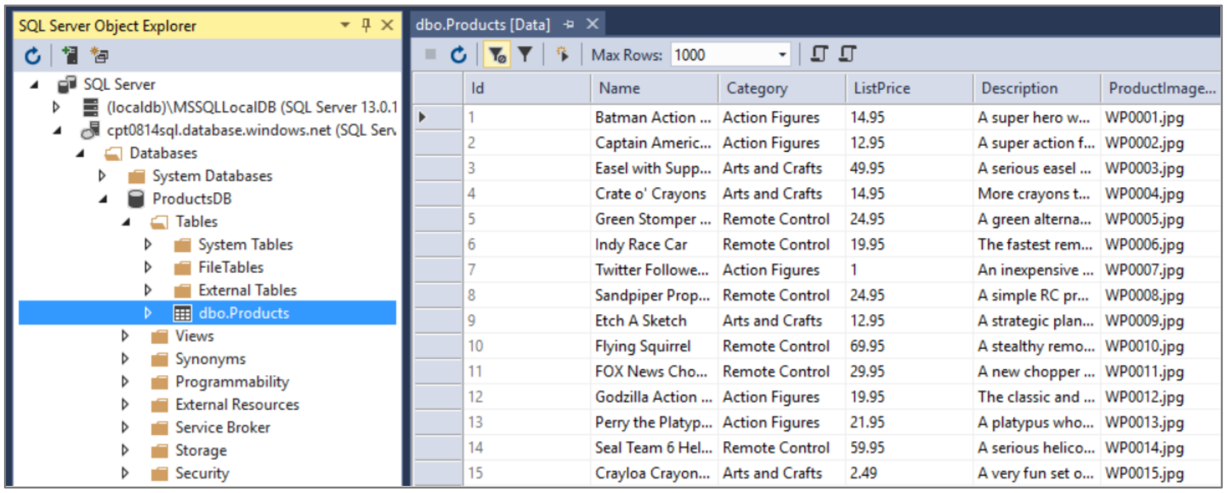

Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Attach...
Server Explorer Toolbox
SQL Server Object Explorer
SQL Server
(localdb)\MSSQLLocalDB (SQL Server 13.0.1601.5 - VM-CBD365\Student)
cpt0814sql.database.windows.net (SQL Server 12.0.2000 - CptStudent, ProductsDB)
Databases
System Databases
ProductsDB
Security
Projects

g) In the **SQL Server Object Explorer**, expand the **ProductsDB** node and then expand the **Tables** node.

h) Right-click on the **dbo.Products** table and select the **View Data** menu command.



i) You should be able to see the data from the **Products** table.

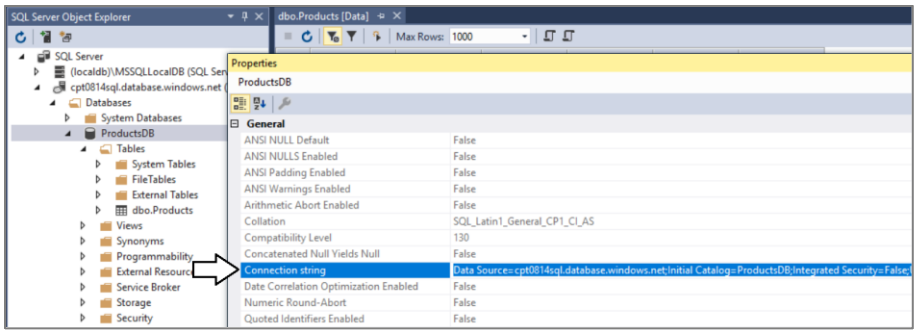


Id	Name	Category	ListPrice	Description	ProductImage...
1	Batman Action ...	Action Figures	14.95	A super hero w...	WP0001.jpg
2	Captain Americ...	Action Figures	12.95	A super action f...	WP0002.jpg
3	Easel with Supp...	Arts and Crafts	49.95	A serious easel ...	WP0003.jpg
4	Crate o' Crayons	Arts and Crafts	14.95	More crayons t...	WP0004.jpg
5	Green Stomper ...	Remote Control	24.95	A green alterna...	WP0005.jpg
6	Indy Race Car	Remote Control	19.95	The fastest rem...	WP0006.jpg
7	Twitter Followe...	Action Figures	1	An inexpensive ...	WP0007.jpg
8	Sandpiper Prop...	Remote Control	24.95	A simple RC pr...	WP0008.jpg
9	Etch A Sketch	Arts and Crafts	12.95	A strategic plan...	WP0009.jpg
10	Flying Squirrel	Remote Control	69.95	A stealthy remo...	WP0010.jpg
11	FOX News Cho...	Remote Control	29.95	A new chopper ...	WP0011.jpg
12	Godzilla Action ...	Action Figures	19.95	The classic and ...	WP0012.jpg
13	Perry the Platyp...	Action Figures	21.95	A platypus who...	WP0013.jpg
14	Seal Team 6 Hel...	Remote Control	59.95	A serious helico...	WP0014.jpg
15	Crayloa Crayon...	Arts and Crafts	2.49	A very fun set o...	WP0015.jpg

j) Close the page which is displaying the data from the **Products** table.

k) In the **SQL Server Object Explorer**, right-click the **ProductsDB** node and then select **Properties**.

l) Examine the database properties such as **Connection String** in the **Properties** dialog.



ANSI NULL Default	False
ANSI NULLS Enabled	False
ANSI Padding Enabled	False
ANSI Warnings Enabled	False
Arithmetic Abort Enabled	False
Collation	SQL_Latin1_General_CP1_CI_AS
Compatibility Level	130
Concatenated Null Yields Null	False
Connection string	Data Source=cpt0814sql.database.windows.net;Initial Catalog=ProductsDB;Integrated Security=False;
Date Correlation Optimization Enabled	False
Numeric Round-Abort	False
Quoted Identifiers Enabled	False

At this point, you have created a new database. In the next exercise, you will begin to develop against this database in an ASP.NET MVC project similar to the project you worked with in the previous lab.

Exercise 2: Connecting to an Azure SQL Database using Entity Framework

In this exercise, you will begin with a pre-provided Visual Studio project named **ProductManagerSQL** which is similar to the project you worked on in the previous lab named **ProductManagerMVC**. You will extend the **ProductManagerSQL** project by adding an Entity Framework model and a strongly-typed controller class to provide read/write access to the **ProductsDB** database.

1. Copy the starter project named **ProductManagerSQL**.

a) In Windows Explorer, navigate to the following folder.

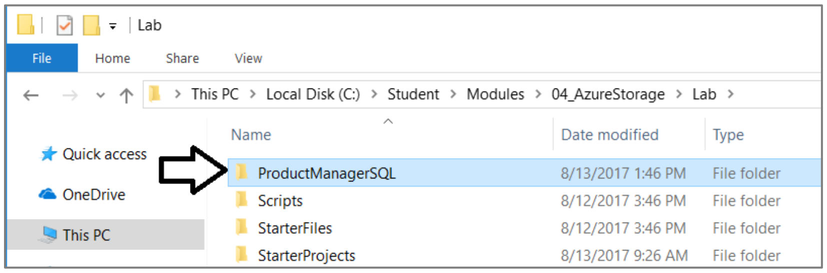
```
C:\Student\Modules\04_AzureStorage\Lab\StarterProjects
```

b) Select the folder named **ProductManagerSQL** and copy it to the windows clipboard.

c) In Windows Explorer, move out one level to the Lab folder at the following location.

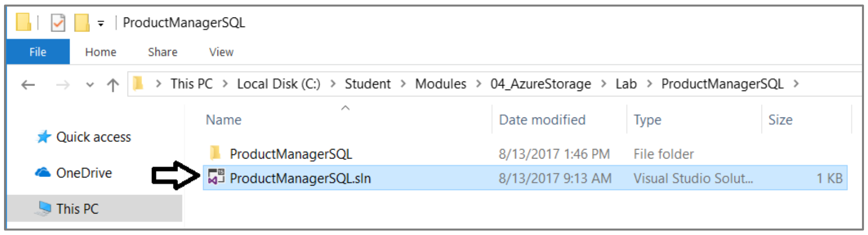
```
C:\Student\Modules\04_AzureStorage\Lab
```

d) Paste the **ProductManagerSQL** folder to make a copy of it inside the **Lab** folder.


Lab
File
Home
Share
View
This PC > Local Disk (C:) > Student > Modules > 04_AzureStorage > Lab >
Quick access
OneDrive
This PC

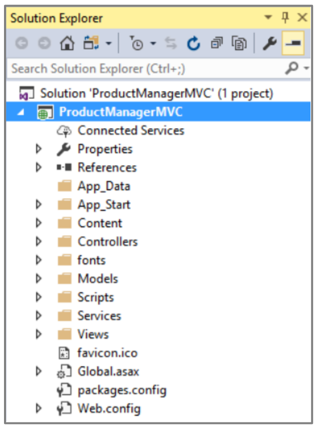
Name	Date modified	Type
ProductManagerSQL	8/13/2017 1:46 PM	File folder
Scripts	8/12/2017 3:46 PM	File folder
StarterFiles	8/12/2017 3:46 PM	File folder
StarterProjects	8/13/2017 9:26 AM	File folder

e) Navigate inside the **ProductMangerSQL** folder and double-click on the solution file named **ProductManagerSQL.sln** to open the **ProductManagerSQL** project in Visual Studio.


ProductManagerSQL
File
Home
Share
View
This PC > Local Disk (C:) > Student > Modules > 04_AzureStorage > Lab > ProductManagerSQL >
Quick access
OneDrive
This PC

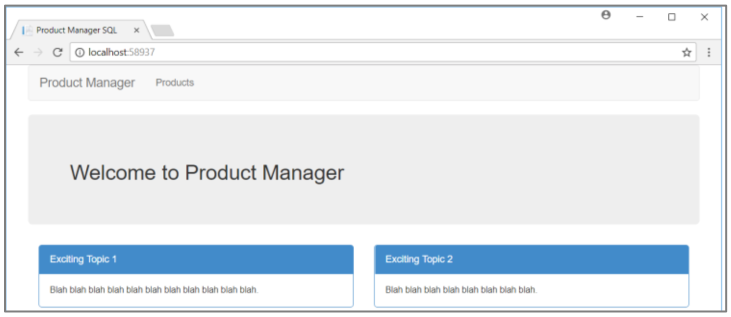
Name	Date modified	Type	Size
ProductManagerSQL	8/13/2017 1:46 PM	File folder	
ProductManagerSQL.sln	8/13/2017 9:13 AM	Visual Studio Solut...	1 KB

f) Take a moment to review the project structure of the **ProductManagerSQL** project.



g) Press the **{F5}** key to start a debugging session in Visual Studio.

h) The project should start and display its home page in the browser as shown in the following screenshot.


Product Manager SQL
localhost:58937
Product Manager
Products
Welcome to Product Manager
Exciting Topic 1
Blah blah blah blah blah blah blah blah blah blah.
Exciting Topic 2
Blah blah blah blah blah blah blah blah.

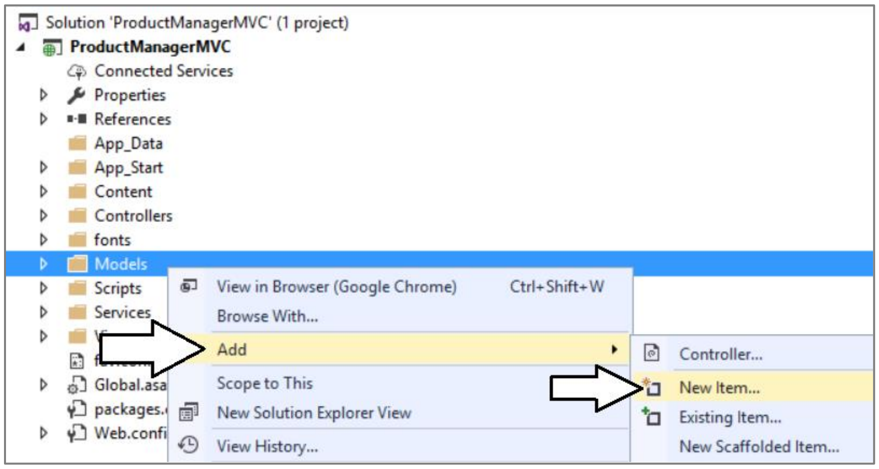
Note the top navbar on the home page contains a **Products** link that does not work correctly yet. Later in this exercise you will add a new **Products** controller so that the **Products** link works correctly.

i) Close the browser and return to Visual Studio and terminate the debugging session.

2. Add a new Entity Framework model to provide access to the **ProductsDB** database.

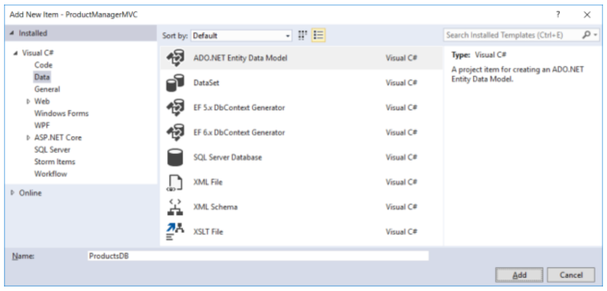
a) In Solution Explorer, locate the top-level folder named **Models**.

b) Right-click the **Models** folder and select the **Add > New Item…** command.



c) In the **Add New Item** dialog, select Visual C# > Data on the left and then select **ADO.NET Entity Data Model**.

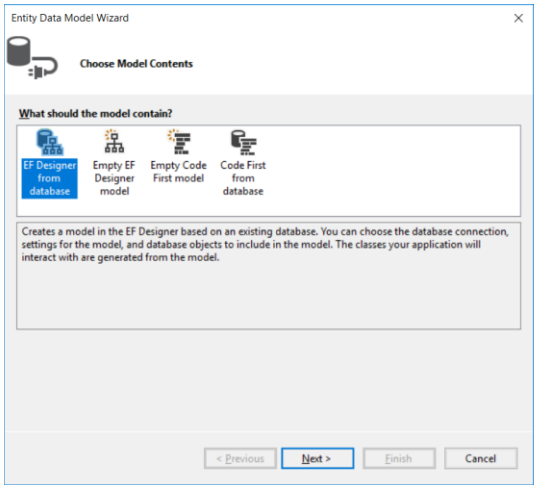
d) In the **Name** textbox, enter a name of **ProductsDB** and then click the **Add** button.


Add New Item - ProductManagerMVC

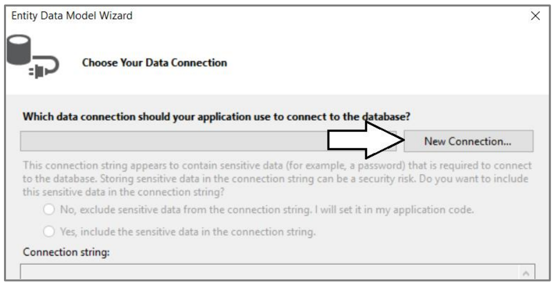
ADO.NET Entity Data Model	Visual C#
DataSet	Visual C#
EF 5.x DbContext Generator	Visual C#
EF 6.x DbContext Generator	Visual C#
SQL Server Database	Visual C#
XML File	Visual C#
XML Schema	Visual C#
XSLT File	Visual C#

Type: Visual C#
A project item for creating an ADO.NET Entity Data Model.
Name: ProductsDB

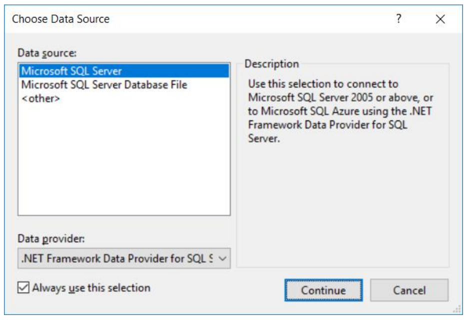
e) On the first page of the **Entity Data Model Wizard**, select **EF Designer from database** and then click **Next**.


Entity Data Model Wizard
Choose Model Contents
What should the model contain?
Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.

f) On the **Choose Your Data Connection** page of the **Entity Data Model Wizard**, click **New Connection…**.

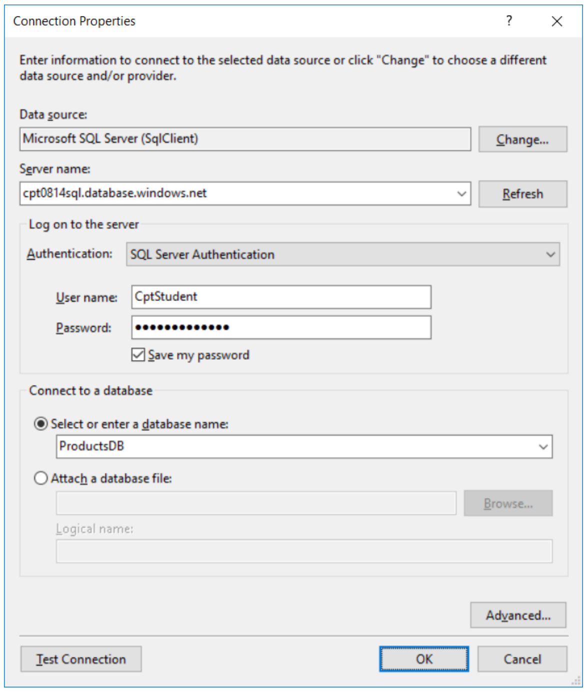

Entity Data Model Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?
No, exclude sensitive data from the connection string. I will set it in my application code.
Yes, include the sensitive data in the connection string.
Connection string:

g) In the **Choose Data Source** dialog, select **Microsoft SQL Server** and click **Continue**.


Choose Data Source
Data source:
Microsoft SQL Server
Microsoft SQL Server Database File
<other>
Description
Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.
Data provider:
.NET Framework Data Provider for SQL S
☑ Always use this selection

h) In the **Connection Properties** dialog, enter the following information.
   i) For **Server name**, add the name of your SQL Server instance including the suffix of **database.windows.net**.
   ii) For **Authentication**, set the value to **SQL Server Authentication**.
   iii) Enter a **User name** of **CptStudent**.
   iv) Enter a **Password** of **pass@word1234**.
   v) Check to **Save my password** checkbox.
   vi) In the **Select or enter a database name** dropdown list, select **ProductsDB**.

i) When the **Connection Properties** dialog matches the following screenshot (except for Server name), click **OK** to continue.



Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: cpt0814sql.database.windows.net

Log on to the server

Authentication: SQL Server Authentication

User name: CptStudent

Password: •••••••••••••

☑ Save my password

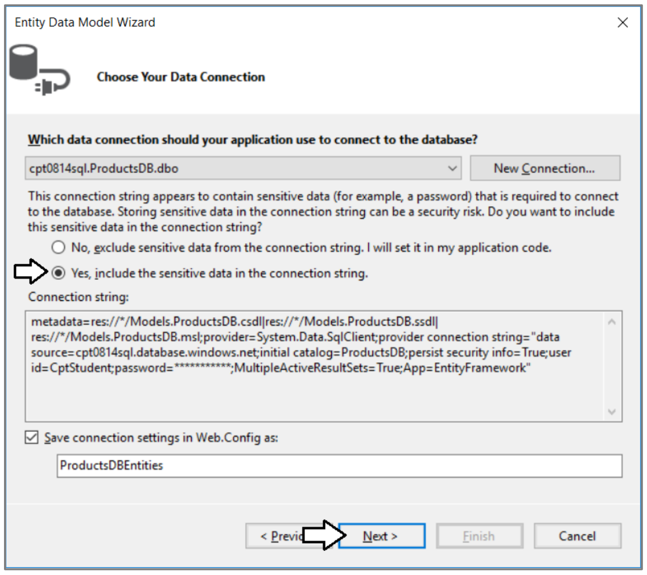
Connect to a database

● Select or enter a database name: ProductsDB

○ Attach a database file:

Logical name:

j) On the **Choose Your Data Connection** page, select the option **Yes, include the sensitive data in the connection string**. and then click **Next** button to continue.



Which data connection should your application use to connect to the database?

cpt0814sql.ProductsDB.dbo

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

○ No, exclude sensitive data from the connection string. I will set it in my application code.

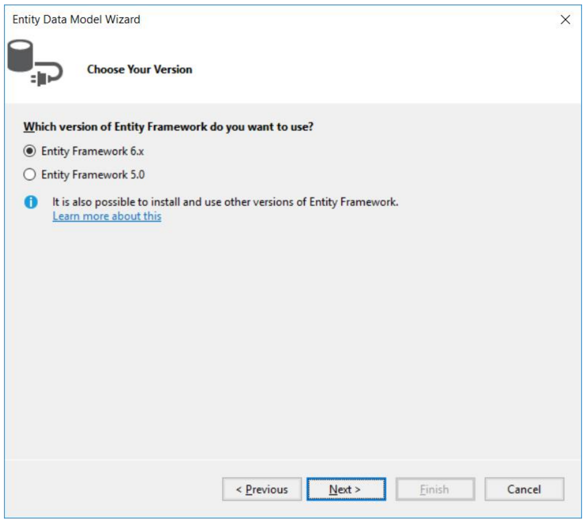
● Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/Models.ProductsDB.csdl|res://*/Models.ProductsDB.ssdl|
res://*/Models.ProductsDB.msl;provider=System.Data.SqlClient;provider connection string="data
source=cpt0814sql.database.windows.net;initial catalog=ProductsDB;persist security info=True;user
id=CptStudent;password=**********;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in Web.Config as: ProductsDBEntities

k) On the **Choose Your Version** dialog, select **Entity Framework 6.x** and click **Next** to continue.


Entity Data Model Wizard

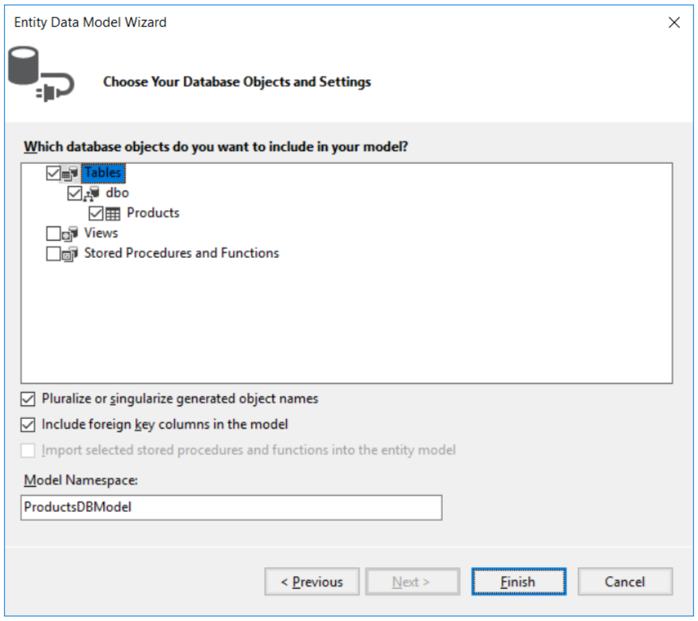
Choose Your Version

Which version of Entity Framework do you want to use?

[✓ Entity Framework 6.x]
Entity Framework 5.0

It is also possible to install and use other versions of Entity Framework.
Learn more about this

l) In the **Choose Your Database Objects and Setting** page, select the **Products** table and click **Next**.


Entity Data Model Wizard

Choose Your Database Objects and Settings

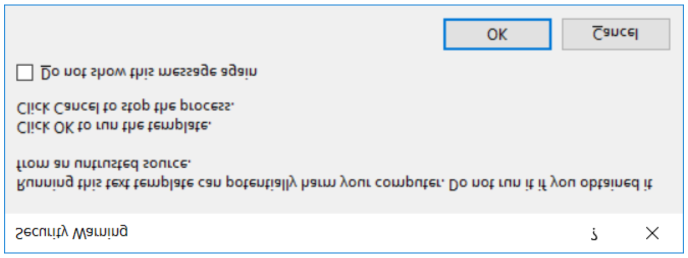
Which database objects do you want to include in your model?

- ☑ Tables
  - ☑ dbo
    - ☑ Products
- ☐ Views
- ☐ Stored Procedures and Functions

☑ Pluralize or singularize generated object names
☑ Include foreign key columns in the model
☐ Import selected stored procedures and functions into the entity model

Model Namespace: ProductsDBModel

m) If you are prompted with a **Security Warning** dialog shown in the following screenshot, click **OK** to continue.


Security Warning

Running this text template can potentially harm your computer. Do not run it if you obtained it from an untrusted source.

Click OK to run the template.
Click Cancel to stop the process.

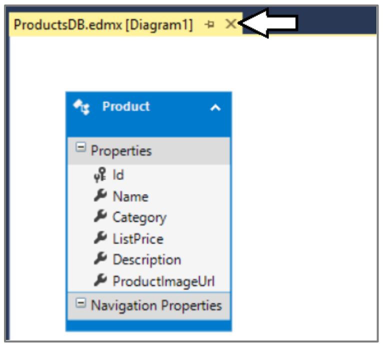
☐ Do not show this message again

It will usually take Visual Studio 30-60 seconds to complete its work building a small Entity Framework model.

n) Wait until Visual Studio completes its work creating the files for the new Entity Framework model.

o) When Visual Studio finishes, it will display a visual designer of the model in a file named **ProductsDB.edmx**.

p) Click the **x** button to close the viewer for **ProductsDB.edmx**.



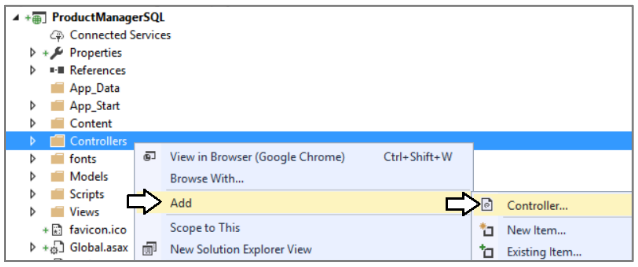
When you create an Entity Model like this, not all the required code is generated until you build and run the project. In the nexg step you will run and build the project one time to ensure all the Entity Framework code is fully built out.

3. Run the project one time to fully build out the Entity Framework model code.
   a) Press the **{F5}** key to start a debugging session.
   b) Once the browser starts and display the home page, close the browser.
   c) Return to Visual Studio and terminate the debugging session.

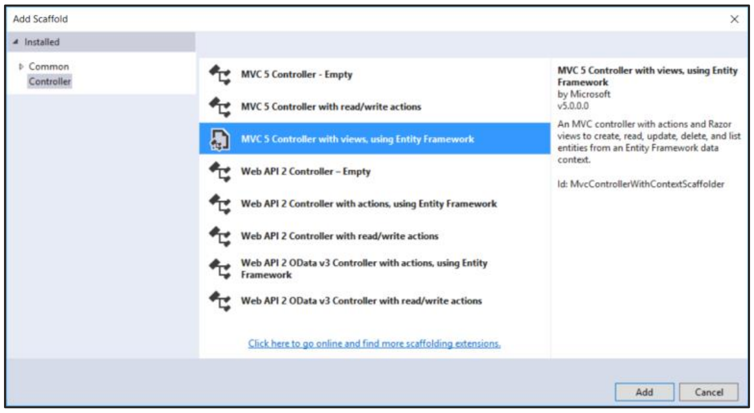
Exercise 3: Extend an MVC Web Application with CRUD Behavior

In this exercise, you will create a strongly-typed controller class using the Entity Framework model you created in an earlier exercise.

1. Add a new strongly-typed controller class named **Products**.
   a) Inside **Solution Explorer**, right-click the **Controllers** folder and then select the **Add > Controller…** menu command.



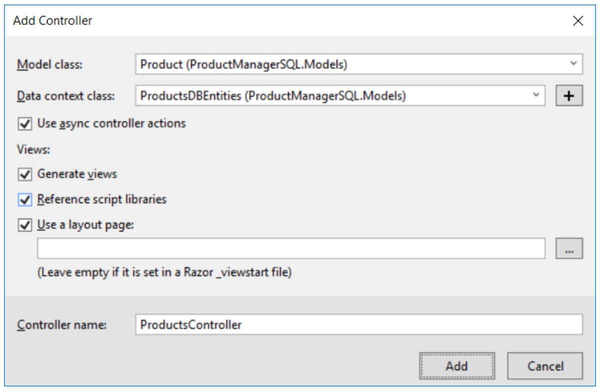
   b) In the **Add Scaffold** dialog, select **MVC 5 Controller with views using Entity Framework** and then click **Add**.


Add Scaffold
Installed
Common
Controller
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData v3 Controller with actions, using Entity Framework
Web API 2 OData v3 Controller with read/write actions
Click here to go online and find more scaffolding extensions.
Add
Cancel

MVC 5 Controller with views, using Entity Framework
by Microsoft
v5.0.0.0
An MVC controller with actions and Razor views to create, read, update, delete, and list entities from an Entity Framework data context.
Id: MvcControllerWithContextScaffolder

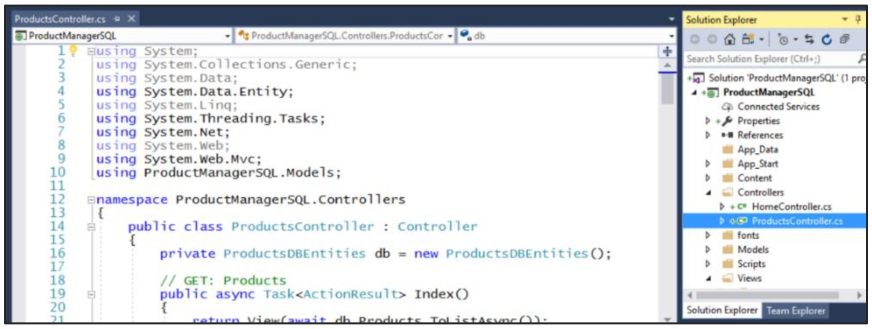
c) In the **Add Controller** dialog, enter the following entries.
   i) Set **Model class** to **Product**.
   ii) Set **Data context class** to **ProductsDBEntities**.
   iii) Check the **Use async controller actions** checkbox.
   iv) Make sure all other checkboxes are checked.
   v) Leave the default value of the **Controller name** which is **ProductsController**.

d) When the **Add Controller** dialog matches the following screenshot, click **Add** to create the new controller class.


Add Controller

Model class: Product (ProductManagerSQL.Models)
Data context class: ProductsDBEntities (ProductManagerSQL.Models)
☑ Use async controller actions
Views:
☑ Generate views
☑ Reference script libraries
☑ Use a layout page:
(Leave empty if it is set in a Razor _viewstart file)
Controller name: ProductsController

e) Visual Studio will create add a new source file named **ProductsController.cs**.

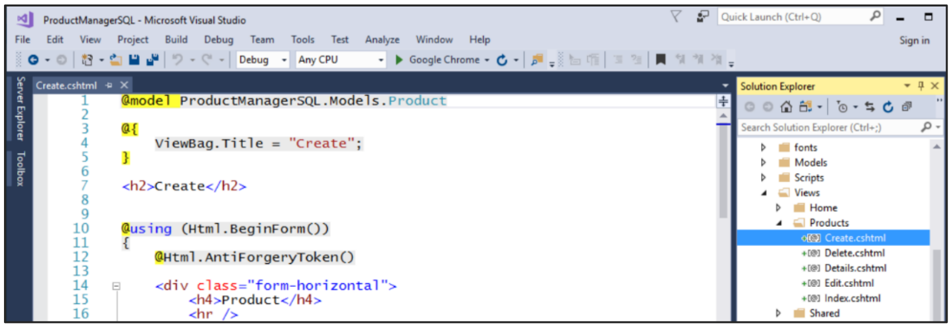


```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Threading.Tasks;
using System.Net;
using System.Web;
using System.Web.Mvc;
using ProductManagerSQL.Models;

namespace ProductManagerSQL.Controllers
{
    public class ProductsController : Controller
    {
        private ProductsDBEntities db = new ProductsDBEntities();

        // GET: Products
        public async Task<ActionResult> Index()
        {
```

f) Take a moment and inspect the method that are defined inside the **ProductsController** class including **Index**, **Details**, **Create**, **Edit**, **Delete** and **DeleteConfirmed**.

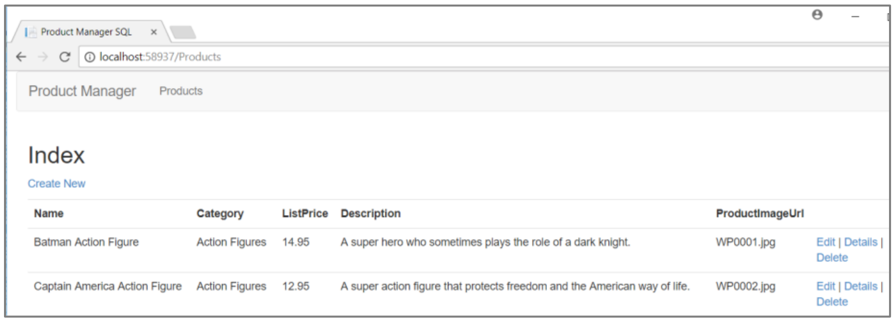
You will notice there are two implementations of the **Create** method and two implementations of the **Edit** method. Each method has one implementation that is execute in response to an GET request which return an input form to the user. Each method has a second overloaded implementation that is execute in response to POST requests which actually perform the edits against the database.

g) Examine the MVC views in the **View/Products** folder. You will notice that Visual Studio has created views for several of the Products controllers action methods including **Create**, **Delete**, **Details**, **Edit** and **Index**.



```
@model ProductManagerSQL.Models.Product

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Product</h4>
        <hr />
```

2. Test out the application by running it and testing the Products controller.
   a) Press the {F5} key to begin a debugging sessions.
   b) When the application starts, click the **Products** link on the top navbar to navigate to the **Products** controller.
   c) You should see the data from the **ProductsDB** database as shown in the following screenshot.



Index

Create New

Name	Category	ListPrice	Description	ProductImageUrl	
Batman Action Figure	Action Figures	14.95	A super hero who sometimes plays the role of a dark knight.	WP0001.jpg	Edit | Details | Delete
Captain America Action Figure	Action Figures	12.95	A super action figure that protects freedom and the American way of life.	WP0002.jpg	Edit | Details | Delete

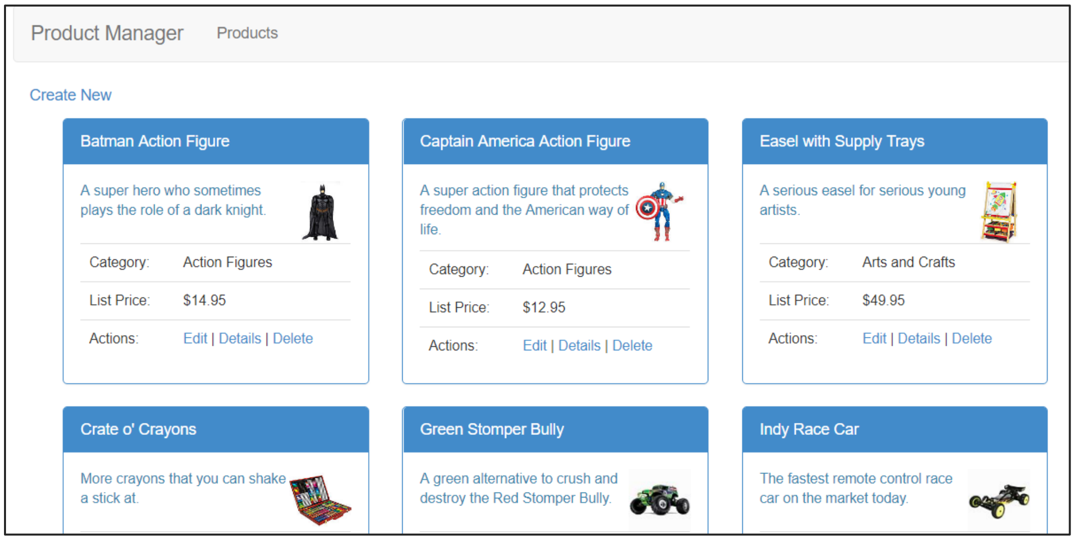
3. Update the code in the MVC view file named **Index.cshtml**.
   a) Return to Visual Studio and update **Index.cshtml** by replacing its contents with the following code.

```
@model IEnumerable<ProductManagerSQL.Models.Product>

<p>
    @Html.ActionLink("Create New", "Create")
</p>

<div class="col-xs-12 col-sm-10">
  @foreach (var product in Model) {
    string productImageUrl = "../Content/ProductImages/" + product.ProductImageUrl;
    <div class="col-lg-4 col-md-5 col-sm-6">
      <div class="panel panel-primary productPanel">
        <div class="panel-heading">@product.Name</div>
        <div class="panel-body">
          <img src="@productImageUrl" class="product-image" />
          <p class="text-info">@product.Description</p>
          <div class="" style="clear: both;">
            <table class="table">
              <tr>
                <td>Category:</td>
                <td>@product.Category</td>
              </tr>
              <tr>
                <td>List Price:</td>
                <td>@product.ListPrice.ToString("$0.00")</td>
              </tr>
              <tr>
                <td>Actions:</td>
                <td>
                  @Html.ActionLink("Edit", "Edit", new { id = product.Id }) |
                  @Html.ActionLink("Details", "Details", new { id = product.Id }) |
                  @Html.ActionLink("Delete", "Delete", new { id = product.Id })
                </td>
              </tr>
            </table>
          </div>
        </div>
      </div>
    </div>
  }
</div>
```

If it is easier, you can copy=and-paste this code from a file in the **Lab\StarterFiles** folder named **Index.cshtml.txt**.

   b) Save your changes to **Index.cshtml**.

4. Test out the application by running it and testing the Products controller.
   a) Press the {F5} key to begin a debugging sessions.
   b) When the application starts, click the **Products** link on the top navbar to navigate to the **Products** controller.
   c) You should see the product data shown in a new layout.



Batman Action Figure
A super hero who sometimes plays the role of a dark knight.

Category:	Action Figures
List Price:	$14.95
Actions:	Edit | Details | Delete

Captain America Action Figure
A super action figure that protects freedom and the American way of life.

Category:	Action Figures
List Price:	$12.95
Actions:	Edit | Details | Delete

Easel with Supply Trays
A serious easel for serious young artists.

Category:	Arts and Crafts
List Price:	$49.95
Actions:	Edit | Details | Delete

Crate o' Crayons
More crayons that you can shake a stick at.

Green Stomper Bully
A green alternative to crush and destroy the Red Stomper Bully.

Indy Race Car
The fastest remote control race car on the market today.

5. Test out he functionality of the **Products** controller.
   a) Click the **Create New** link at the top of the **Products** page and make sure you can add a new product.
   b) Click on the **Edit** link in an existing product and make sure you can modify existing product data.
   c) Click on the **Delete** link and make sure you can delete and existing product.

Congratulations. You have now completed this lab.