**Milestone #2 - A***Worth 3%*

*(Due Tuesday October 20th, 2020 – at midnight*

*Late policy: 10%/day, maximum 2 days)*

**Overview**

**=========**

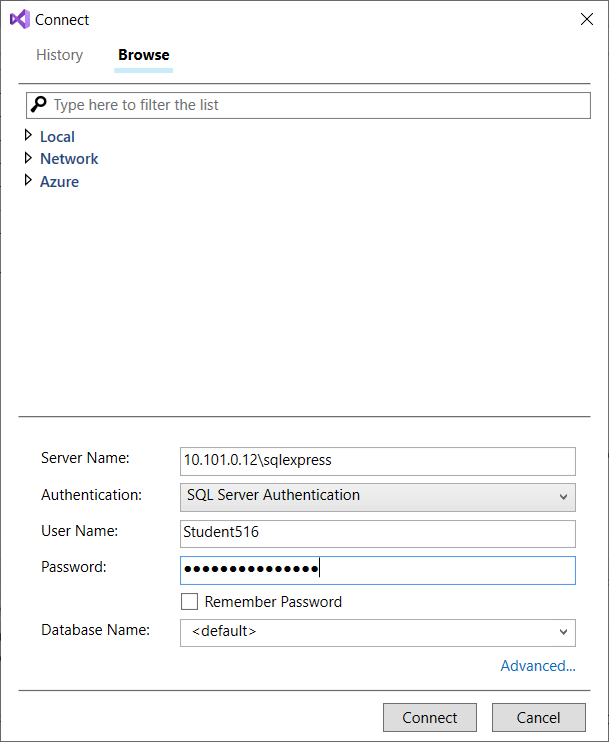
Everyone in your team must create an ASP.NET Core 3 MVC web application for a fictitious bicycle manufacturer. For this Milestone, you will be using Entity Framework Core to access the database. Install/Update Visual Studio 2019 to at least version 16.3 in order to support .Net Core 3 applications.

**Database**

**=========**

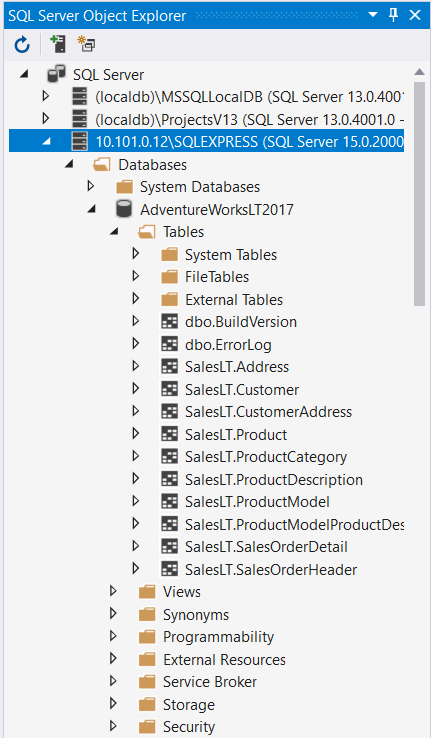
We will be using the AdventureWorksLT2017 MSSQL database. The AdventureWorks LT database supports standard online transaction processing scenarios for a fictitious bicycle manufacturer.

You can connect to the DB in two ways: 1) Using SQL server Management Studio (as in the *Entity\_Framework\_Core \_setup.docx tutorial)* or 2) using Visual Studio as in the steps below:



1. In Visual Studio, click View 🡪 SQL Server Object Explorer.
2. Right click on SQL Server 🡪 Add SQL Server
3. Accessing the DB from the college:
   1. If you are home, you need to connect to the college’s VPN, before connecting to the college server.
   2. Temporary password is: SecretSauce2020
   3. Every team will be provided with their own DB and username/password (later on)

After connecting, you should see:



Take the time to analyze the **tables Product and ProductCategory,** the **vProductAndDescription view** and the data they return.

**Database-First approach**

**=========**

Reverse engineer the database tables and views to C# Classes.

**IMPORTANT**: First complete the *Entity\_Framework\_Core \_setup.docx* documentation on Lea.

**Create BikesController.cs**

**=========**

Create the controller BikesController.cs based on the table ProductCategory

The Bikes controller shows dynamically the model of bikes it has.

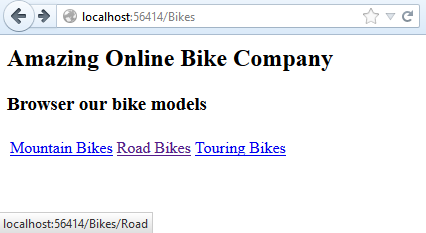
public IActionResult Index() { … }

Convert the following SQL to LINQ to use it against the EntityFramework:

SELECT \*

FROM SalesLT.ProductCategory

WHERE ParentProductCategoryID = 1



public IActionResult Road() { … }

Convert the following SQL to LINQ to use it against the EntityFramework:

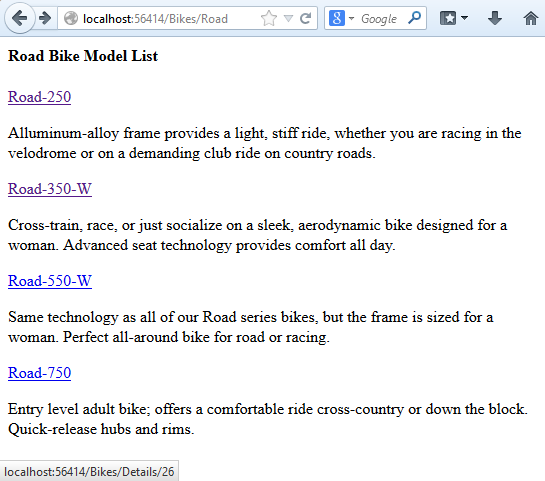
SELECT DISTINCT ProductModel, Description, ProductModelID

FROM Saleslt.vProductAndDescription

WHERE Culture = 'en' AND ProductCategoryID = 6 AND SellEndDate IS NULL

The resulting LINQ is a Select operator (projection) with anonymous type. This is because the LINQ statement is not equivalent to SELECT \*.

See LINQ Select - Anonymous Types examples:   
<https://code.msdn.microsoft.com/LINQ-to-DataSets-09787825/description>



**BikeListModel (used for Road/Mountain/Touring methods)**

**=========**

We need to create this class in order to send a list of BikeListModel to the strongly typed Road view.

namespace Milestone3.Models //the namespace can be different

{

public class BikeListModel : BikeDBContext //the context can be different

{

public string ProductModel { get; set; }

public string Description { get; set; }

public int ProductModelID { get; set; }

}

}

public IActionResult Details(int? id) { … }

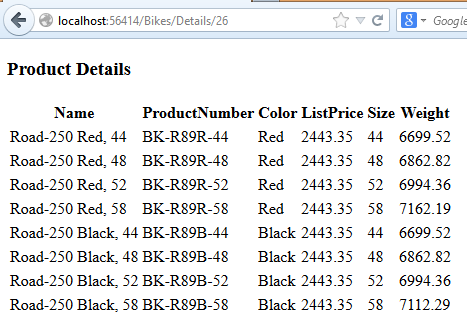
Details action shows product details for the ProductModelID sent.

Convert the following SQL to LINQ to use it against the Entity Framework:

SELECT \*

FROM Saleslt.Product

WHERE ProductModelID = id



**ASP.NET Core 3 MVC Components**

**=========**

* ASP.NET Core 3 MVC: Model/View/Controller   
  <https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/?view=aspnetcore-3.0>
* Entity Framework Core: <https://docs.microsoft.com/en-us/ef/core/>  
  <https://www.entityframeworktutorial.net/efcore/install-entity-framework-core.aspx>
* Controller: BikesController.cs
* Model: BikeListModel.cs
* Views:
  + ~/Bikes/Index.cshtml
  + ~/Bikes/Road.cshtml
  + ~/Bikes/Touring.cshtml
  + ~/Bikes/Mountain.cshtml
  + ~/Bikes/Details.cshtml
* LINQ: <http://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b>

**Important notes about EntityFramework Core**

**=========**

The Entity Framework will automatically create classes from tables but it has trouble converting the database views to classes. When you try to scaffold a class based on a database view, you will receive an error message related to “Primary key not found”

You manually have to modify the class VProductAndDescription.cs based on the DB view:

Set the view’s primary key:

* Add [Key] data annotation only to ProductId. This makes it primary key.
* Only SellEndDate Property should be nullable. Remove the ? making Properties nullable for all except for SellEndDate.

Save and Build (CTRL+SHIFT+B).

**Requirements**

**=========**

**Your application must meet the following requirements:**

1. **It must be implemented in ASP.NET Core 3 MVC with EntityFramework Core in C#.**
2. **Design is graded: You should include a logo for your company and realistic text description in all pages. The more pictures the better.** 
   1. **Make it presentable. Inspire yourself from:** [**http://www.trekbikes.com/ca/en/**](http://www.trekbikes.com/ca/en/)
3. **Prior to submitting, have a teammates test your application.**

**Deliverables**

**=========**

* **Compress the Milestone project folder and submit on Lea. If the zipped file is bigger than 50MB then: use 7zip compression level Ultra and split to volumes of 50 MB each. It will create 2 or more 7zip files. Upload one by one on Lea (limit per file is 50 MB).**

You are to keep your entire project in one folder. Compress this entire folder into a ZIP file (retain the tree structure please!) and submit this on LEA