

## Car Repository - Insert

---

### Description

### Note :

**EFCore.csproj** file is given for your reference in the code template. Do not make any changes here.

**Objective:** This exercise aims to introduce learners to developing a C# application utilizing Entity Framework Core to store car details in a SQL Server database.

### Scenario:

Sam, a skilled software developer, recently joined a prestigious car company. As part of his responsibilities, he has been assigned the task of creating an advanced application for adding car details to the company's database.

Help them to create a **C#** application to store details in the **SQLSERVER** database and use the **entity framework core** to connect to the database, which provides the following functionalities.

### Functionalities:

- Add car details to the database.

### Requirements:

1. In the class **Car**, implement the below given public properties.

Data Type	Property Name
int	Id

string	Brand
string	Model
double	Price

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Car** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Car** class.

2. In the class **Car Context**, implement the below given property, method and also inherit the class **DbContext**.

Data Type	Property Name
DbSet<Car>	Cars

Method	Description
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)	This method is used to configure the database connection for the <b>DbContext</b> .

**appsettings.json:**

```
{
  "ConnectionStrings": {
```

```
"DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial
Catalog=StudentDB;Integrated Security=True;"

}

}
```

This file is already provided for you. Use the connection string in your context class **OnConfiguring** method to configure the database.

The below-given reference code is the part of the CarContext.cs

3. In the class **CarRepository**, implement the below given public method.

Method	Description
public bool AddCar(Car car)	This method is used to add the given car details to the database.  If the car details are added successfully, then return <b>true</b> .

4. In the class **Program - Main()** method,

-- Get the all input values from the user.

-- Call the AddCar method and print **Details Added Successfully.**

**Note:**

- Keep the properties, methods and classes as **public**.
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

**Sample Input/output:**

Enter car id

**1**

Enter car brand

**Hyundai**

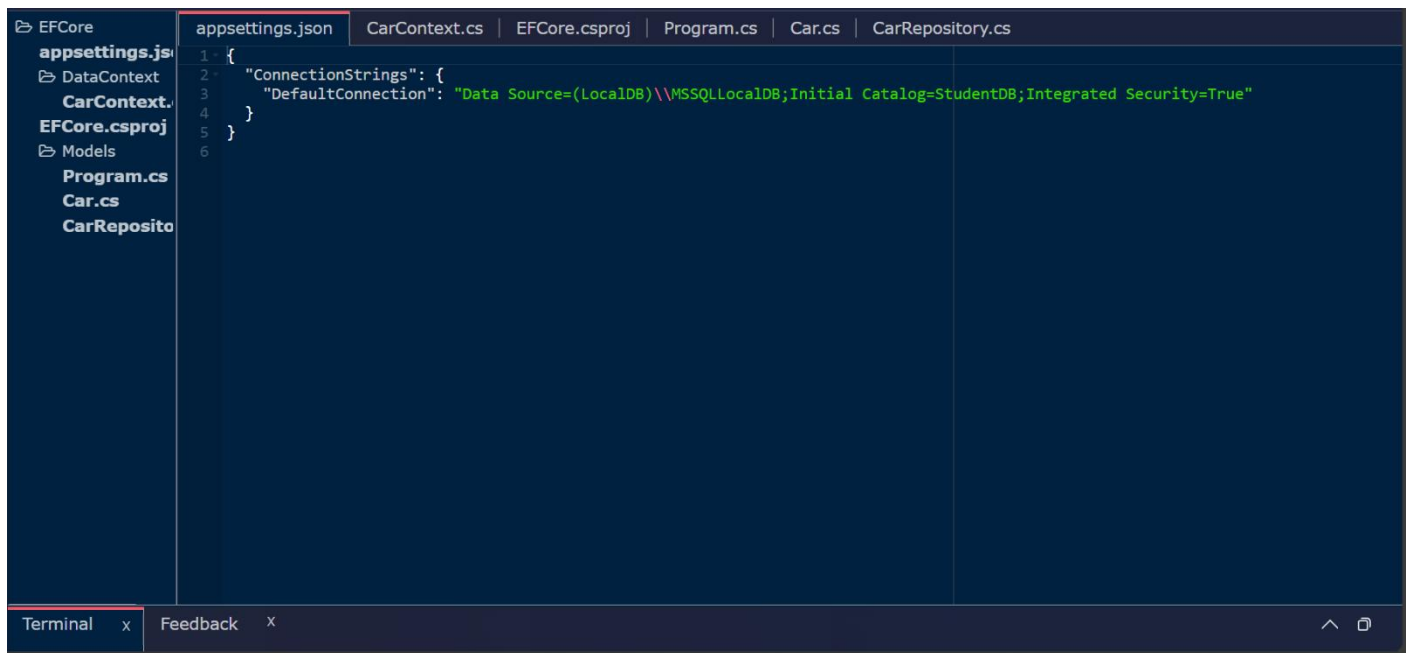
Enter car model

**Hyundai Elantra**

Enter car price

**1800000**

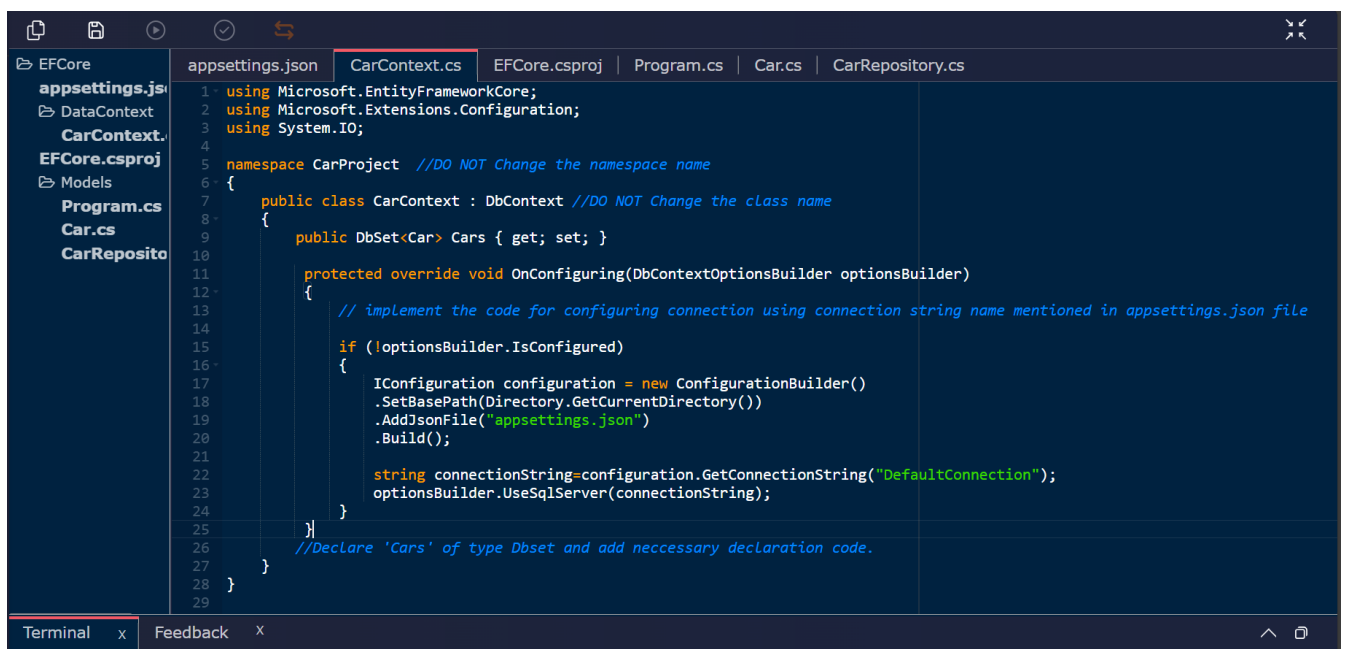
Details Added Successfully



The screenshot shows the Visual Studio IDE with the EFCore project selected in the Solution Explorer. The file explorer on the left lists the following files: appsettings.json, DataContex, CarContext., EFCore.csproj, Models, Program.cs, Car.cs, and CarRepository. The main editor window displays the content of appsettings.json, which is a JSON object with a "ConnectionStrings" property. The code is as follows:

```
1 {
2   "ConnectionStrings": {
3     "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True"
4   }
5 }
6
```

The bottom of the window shows a "Terminal" tab and a "Feedback" button.



The screenshot shows the Visual Studio IDE with the EFCore project selected in the Solution Explorer. The file explorer on the left lists the following files: appsettings.json, DataContex, CarContext., EFCore.csproj, Models, Program.cs, Car.cs, and CarRepository. The main editor window displays the content of CarContext.cs, which is a C# class that inherits from DbContext. The code is as follows:

```
1 using Microsoft.EntityFrameworkCore;
2 using Microsoft.Extensions.Configuration;
3 using System.IO;
4
5 namespace CarProject //DO NOT Change the namespace name
6 {
7   public class CarContext : DbContext //DO NOT Change the class name
8   {
9     public DbSet<Car> Cars { get; set; }
10
11     protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
12     {
13       // implement the code for configuring connection using connection string name mentioned in appsettings.json file
14
15       if (!optionsBuilder.IsConfigured)
16       {
17         IConfiguration configuration = new ConfigurationBuilder()
18           .SetBasePath(Directory.GetCurrentDirectory())
19           .AddJsonFile("appsettings.json")
20           .Build();
21
22         string connectionString=configuration.GetConnectionString("DefaultConnection");
23         optionsBuilder.UseSqlServer(connectionString);
24       }
25     }
26     //Declare 'Cars' of type Dbset and add necessary declaration code.
27   }
28 }
29
```

The bottom of the window shows a "Terminal" tab and a "Feedback" button.

cognizant.tekstac.com says

Action Required: This action is not allowed in the editor.

EFCore | appsettings.json | CarContext.cs | EFCore.csproj | Program.cs | Car.cs | CarRepository.cs

appsettings.json

```
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>net6.0</TargetFramework>
6     <ImplicitUsings>enable</ImplicitUsings>
7     <Nullable>enable</Nullable>
8     <GenerateProgramFile>>false</GenerateProgramFile>
9   </PropertyGroup>
10
11   <ItemGroup>
12     <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0-preview.4.23259.3" />
13     <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0-preview.4.23259.3">
14       <PrivateAssets>all</PrivateAssets>
15       <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
16     </PackageReference>
17     <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="8.0.0-preview.4.23259.5" />
18     <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.6.0" />
19     <PackageReference Include="NUnit" Version="3.13.3" />
20     <PackageReference Include="NUnit3TestAdapter" Version="4.5.0" />
21   </ItemGroup>
22
23   <ItemGroup>
24     <None Update="appsettings.json">
25       <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
26     </None>
27   </ItemGroup>
28 </Project>
29
```

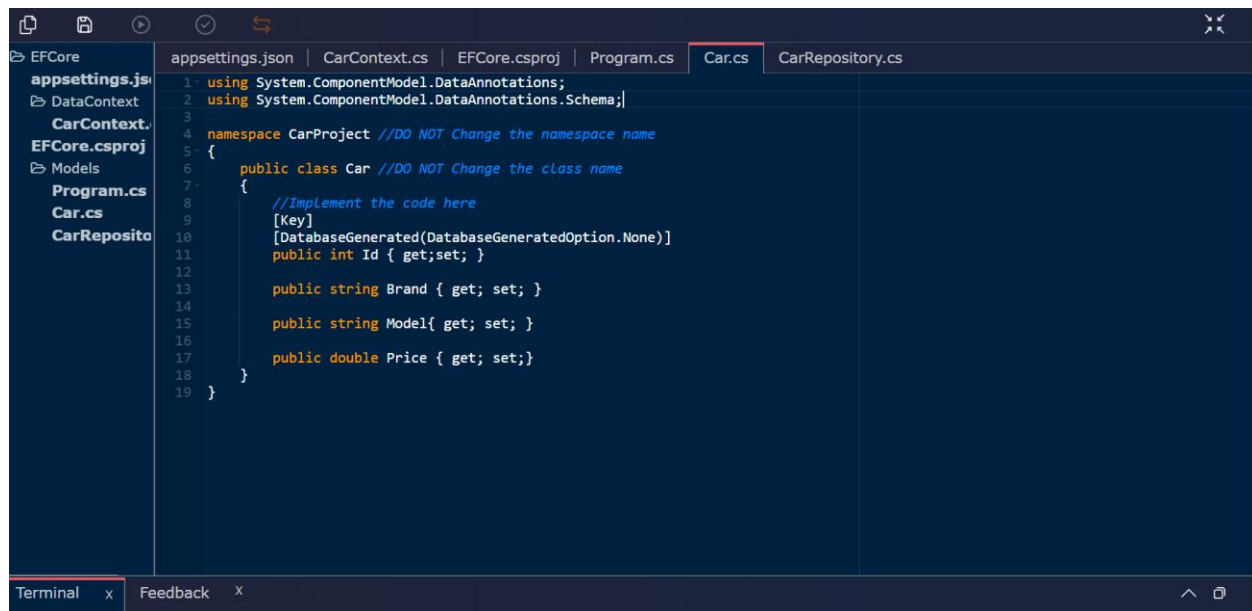
Terminal x Feedback x

EFCore | appsettings.json | CarContext.cs | EFCore.csproj | Program.cs | Car.cs | CarRepository.cs

Program.cs

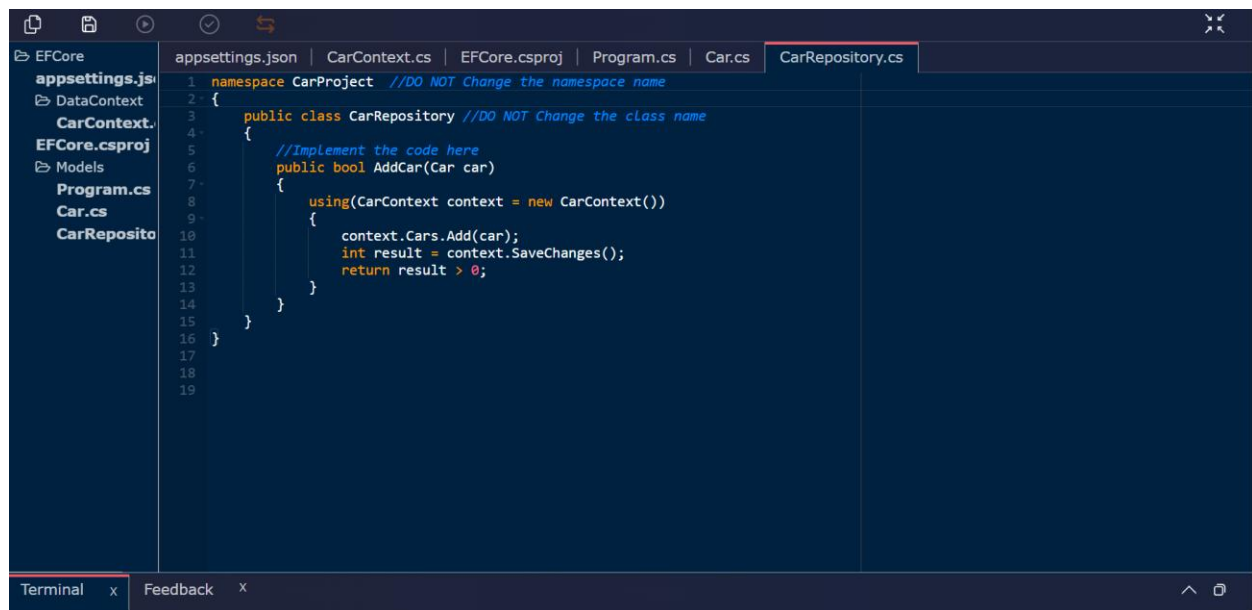
```
1 using System;
2 namespace CarProject //DO NOT Change the namespace name
3 {
4   public class Program //DO NOT Change the class name
5   {
6     public static void Main(string[] args)
7     { //Implement the code here
8       Console.WriteLine("Enter car id");
9       int id=Convert.ToInt32(Console.ReadLine());
10      Console.WriteLine("Enter car brand");
11      string brand = Console.ReadLine();
12      Console.WriteLine("enter car model");
13      string model = Console.ReadLine();
14      Console.WriteLine("enter car price");
15      double price= Convert.ToDouble(Console.ReadLine());
16      Car car=new Car
17      {
18        Id=id,
19        Brand = brand,
20        Model=model,
21        Price=price  };
22      CarRepository repo=new CarRepository();
23      bool isAdded= repo.AddCar(car);
24      if (isAdded){
25        Console.WriteLine("Details added Successfully");
26      }
27    }
28  }
29
```

Terminal x Feedback x



Visual Studio interface showing the Car.cs file in the CarProject namespace. The file contains a Car class with properties Id, Brand, Model, and Price.

```
1 using System.ComponentModel.DataAnnotations;
2 using System.ComponentModel.DataAnnotations.Schema;
3
4 namespace CarProject //DO NOT Change the namespace name
5 {
6     public class Car //DO NOT Change the class name
7     {
8         //Implement the code here
9         [Key]
10        [DatabaseGenerated(DatabaseGeneratedOption.None)]
11        public int Id { get; set; }
12
13        public string Brand { get; set; }
14
15        public string Model { get; set; }
16
17        public double Price { get; set; }
18    }
19 }
```



Visual Studio interface showing the CarRepository.cs file in the CarProject namespace. The file contains a CarRepository class with an AddCar method.

```
1 namespace CarProject //DO NOT Change the namespace name
2 {
3     public class CarRepository //DO NOT Change the class name
4     {
5         //Implement the code here
6         public bool AddCar(Car car)
7         {
8             using(CarContext context = new CarContext())
9             {
10                context.Cars.Add(car);
11                int result = context.SaveChanges();
12                return result > 0;
13            }
14        }
15    }
16 }
```

## Car Repository - Eager Loading

---

### Description

#### Note :

**EFCore.csproj** file is given for your reference in the code template. Do not make any changes here.

**Objective:** This exercise aims to introduce learners to the concept of eager loading in Entity Framework Core and demonstrate its usage to optimize data retrieval by efficiently loading related data along with the main entities in a single query.

#### Scenario:

Sam has been assigned the new task of creating an advanced application for retrieving car and make details from the company's database.

Help them create a **C#** application to retrieve details from the **SQLSERVER** database and use the **entity framework core** to connect to the database, which provides the following functionalities.

#### Functionalities:

- Retrieving car and make details from the database using **Eager Loading**.

#### Requirements:

1. In the class **Car**, implement the below given public properties.

Data Type	Property Name
int	Id



string	Model
int	Year
int	MakeId
Make	Make

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Car** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Car** class.

2. In the class **Make**, implement the below given public properties.

Data Type	Property Name
int	Id
string	Name

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Make** class.
- **DatabaseGeneratedOption. None** attribute - which is used to make the **Id** property a non-identity primary key in the **Make** class.

3. In the class **Car Context**, implement the below given property, method and also inherit the class **DbContext**.

Data Type	Property Name
DbSet<Car>	Cars
DbSet<Make>	Makes

Method	Description
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)	This method is used to configure the database connection for the <b>DbContext</b> .

#### appsettings.json:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial
Catalog=StudentDB;Integrated Security=True;"
  }
}
```

This file is already provided for you. Use the connection string in your context class **OnConfiguring** method to configure the database.

The below-given reference code is the part of the CarContext.cs

4. In the class **CarRepository**, implement the below given public method.

Method	Description
<pre>public static IEnumerable&lt;Car&gt; GetAllCarsWithMake(CarContext context)</pre>	<p>This method is used to get the cars with make details from the database.</p> <p><b>Hint:</b> Use <b>Include</b> method (Eager Loading Concept).</p>

4. In the class **Program - Main()** method,

-- Call the GetAllCarsWithMake method and display the result as per the sample output.

5. The "**GetMyExpression**" method is for testing your LINQ QUERY EXPRESSION. So fill your query expression in the space holder provided. ONLY THE QUERY EXPRESSION Nothing more needs to be implemented in this method.

The below sample data is already available in the database table **Makes**.

<b>Id</b>	<b>Name</b>
101	Toyota
102	Ford
103	Honda
104	BMW

The below sample data is already available in the database table **Cars**.

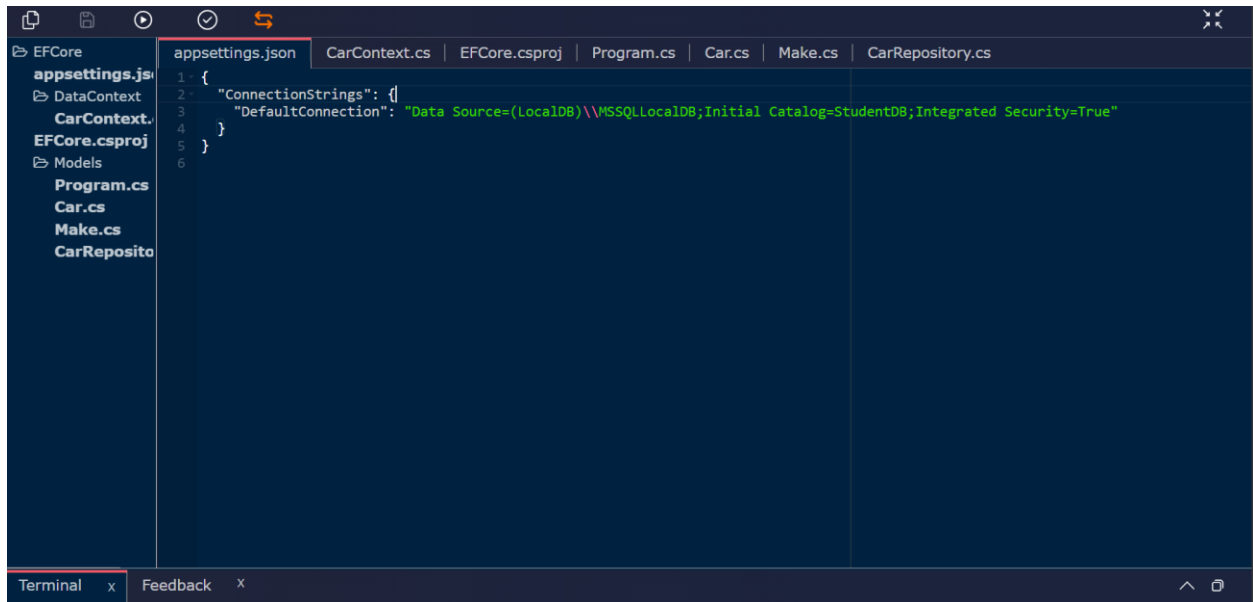
<b>Id</b>	<b>Model</b>	<b>Year</b>	<b>Makeld</b>
1	Camry	1999	101
2	Mustang	2002	102
3	Sienna	2000	101
4	Civic	1998	103
5	X3	2002	104

**Note:**

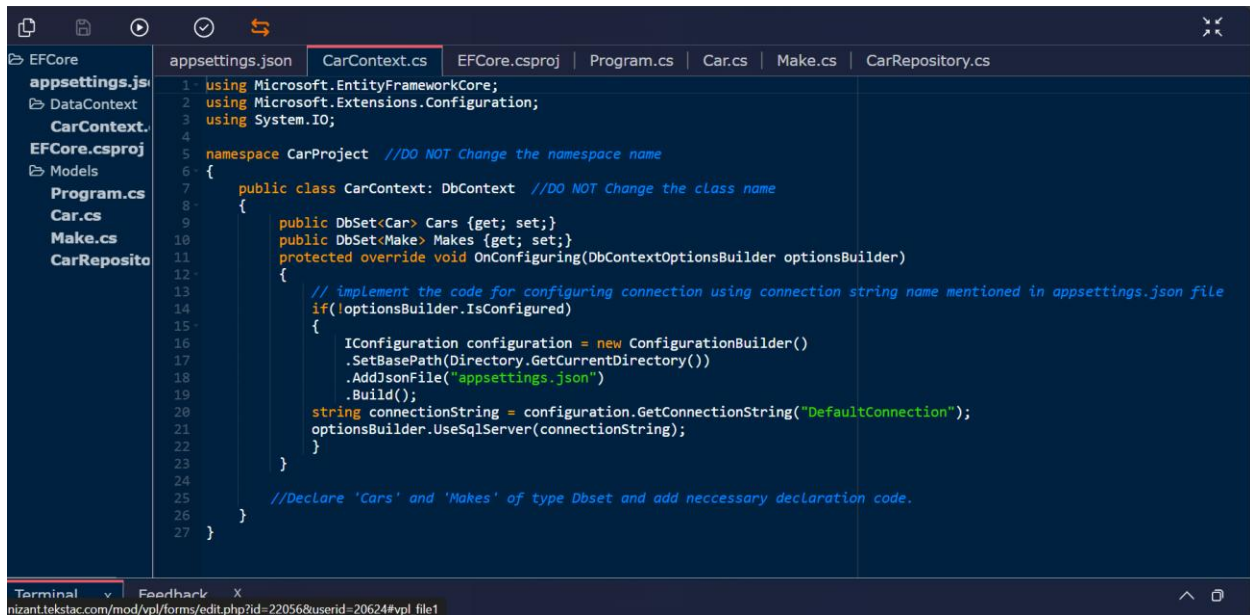
- Keep the properties, methods and classes as **public**.
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

### Sample Output :

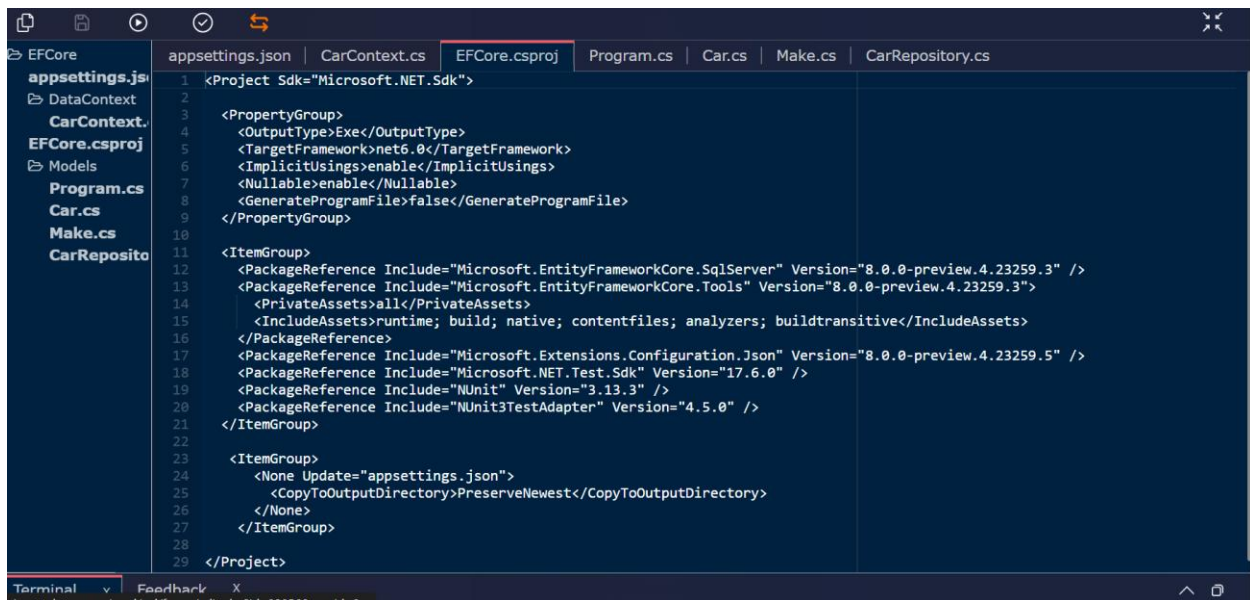
Car Id: 1,    Make: Toyota,    Model: Camry,    Year: 1999  
Car Id: 2,    Make: Ford,    Model: Mustang,    Year: 2002  
Car Id: 3,    Make: Toyota,    Model: Sienna,    Year: 2000  
Car Id: 4,    Make: Honda,    Model: Civic,    Year: 1998  
Car Id: 5,    Make: BMW,    Model: X3,    Year: 2002



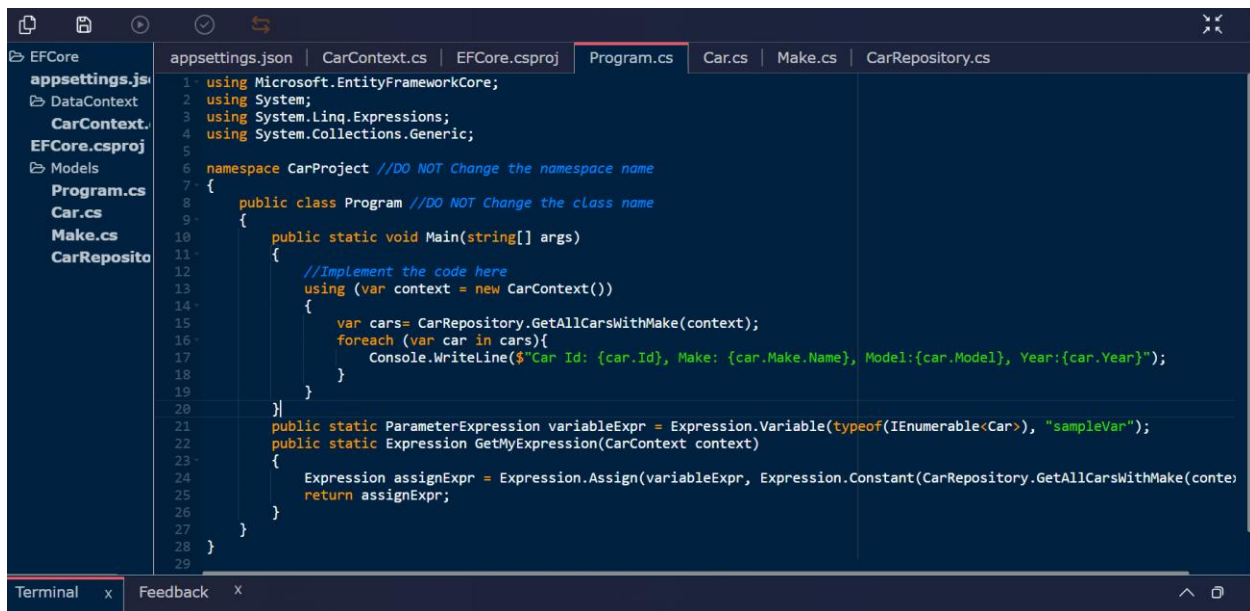
```
1 {  
2   "ConnectionStrings": {  
3     "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True"  
4   }  
5 }  
6
```



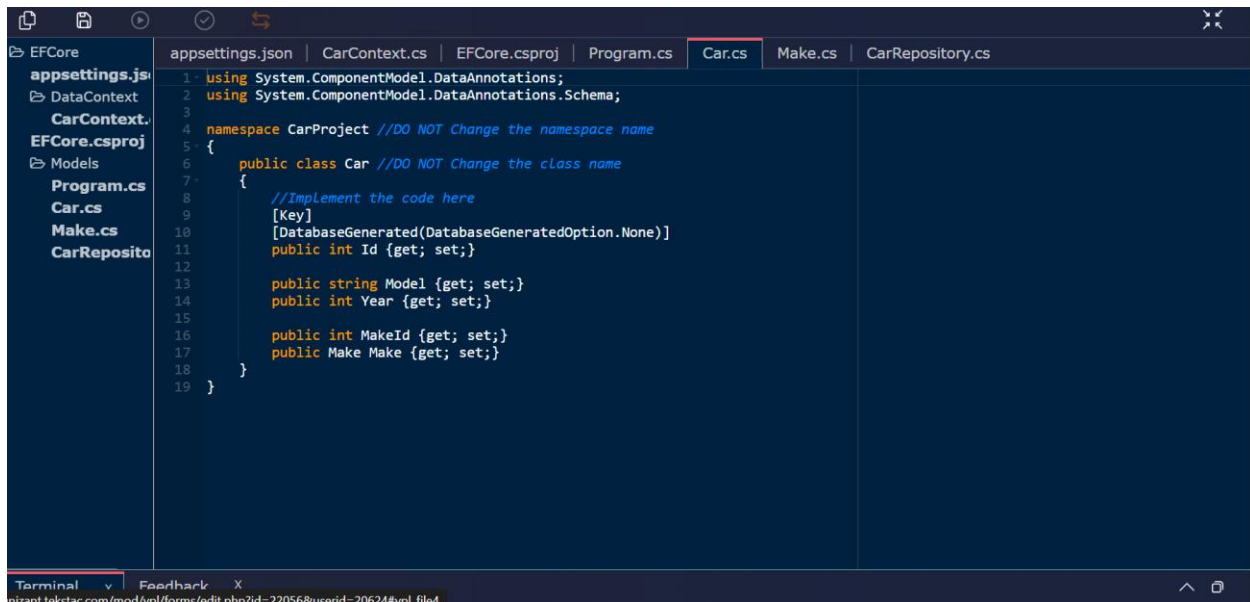
```
1 using Microsoft.EntityFrameworkCore;
2 using Microsoft.Extensions.Configuration;
3 using System.IO;
4
5 namespace CarProject //DO NOT Change the namespace name
6 {
7     public class CarContext: DbContext //DO NOT Change the class name
8     {
9         public DbSet<Car> Cars {get; set;}
10        public DbSet<Make> Makes {get; set;}
11        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
12        {
13            // implement the code for configuring connection using connection string name mentioned in appsettings.json file
14            if(!optionsBuilder.IsConfigured)
15            {
16                IConfiguration configuration = new ConfigurationBuilder()
17                    .SetBasePath(Directory.GetCurrentDirectory())
18                    .AddJsonFile("appsettings.json")
19                    .Build();
20                string connectionString = configuration.GetConnectionString("DefaultConnection");
21                optionsBuilder.UseSqlServer(connectionString);
22            }
23        }
24
25        //Declare 'Cars' and 'Makes' of type DbSet and add necessary declaration code.
26    }
27 }
```



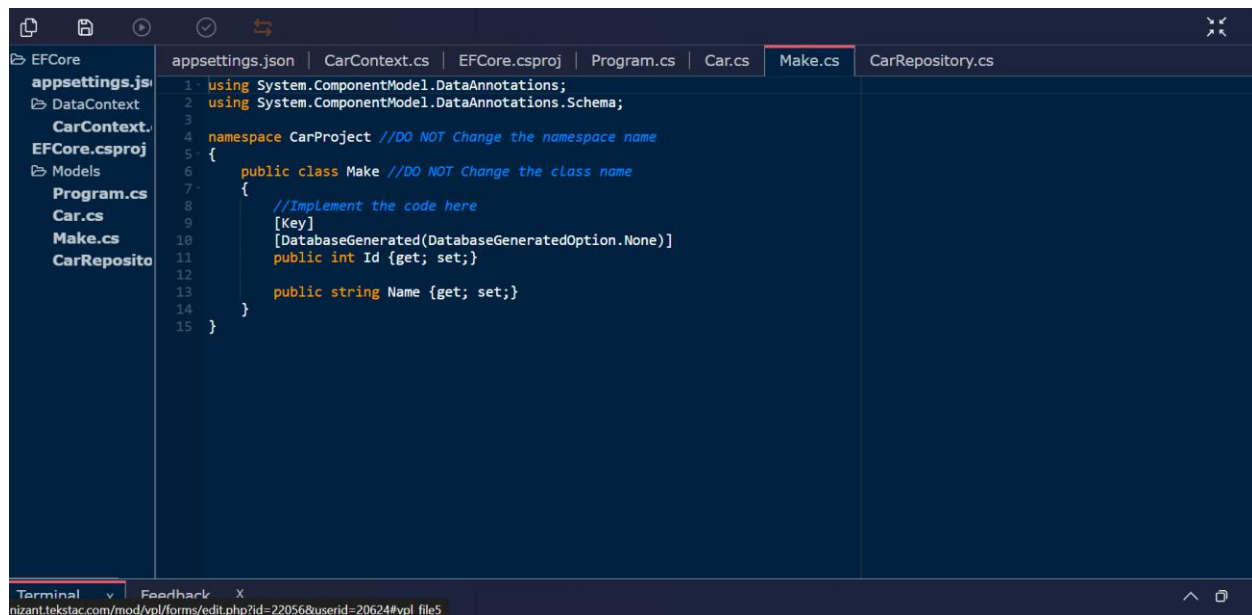
```
1 <Project Sdk="Microsoft.NET.Sdk">
2
3     <PropertyGroup>
4         <OutputType>Exe</OutputType>
5         <TargetFramework>net6.0</TargetFramework>
6         <ImplicitUsings>enable</ImplicitUsings>
7         <Nullable>enable</Nullable>
8         <GenerateProgramFile>>false</GenerateProgramFile>
9     </PropertyGroup>
10
11     <ItemGroup>
12         <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0-preview.4.23259.3" />
13         <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0-preview.4.23259.3">
14             <PrivateAssets>all</PrivateAssets>
15             <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
16         </PackageReference>
17         <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="8.0.0-preview.4.23259.5" />
18         <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.6.0" />
19         <PackageReference Include="NUnit" Version="3.13.3" />
20         <PackageReference Include="NUnit3TestAdapter" Version="4.5.0" />
21     </ItemGroup>
22
23     <ItemGroup>
24         <None Update="appsettings.json">
25             <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
26         </None>
27     </ItemGroup>
28
29 </Project>
```



```
1 using Microsoft.EntityFrameworkCore;
2 using System;
3 using System.Linq.Expressions;
4 using System.Collections.Generic;
5
6 namespace CarProject //DO NOT Change the namespace name
7 {
8     public class Program //DO NOT Change the class name
9     {
10         public static void Main(string[] args)
11         {
12             //Implement the code here
13             using (var context = new CarContext())
14             {
15                 var cars= CarRepository.GetAllCarsWithMake(context);
16                 foreach (var car in cars){
17                     Console.WriteLine($"Car Id: {car.Id}, Make: {car.Make.Name}, Model:{car.Model}, Year:{car.Year}");
18                 }
19             }
20         }
21         public static ParameterExpression variableExpr = Expression.Variable(typeof(IEnumerable<Car>), "sampleVar");
22         public static Expression GetMyExpression(CarContext context)
23         {
24             Expression assignExpr = Expression.Assign(variableExpr, Expression.Constant(CarRepository.GetAllCarsWithMake(context)));
25             return assignExpr;
26         }
27     }
28 }
29
```



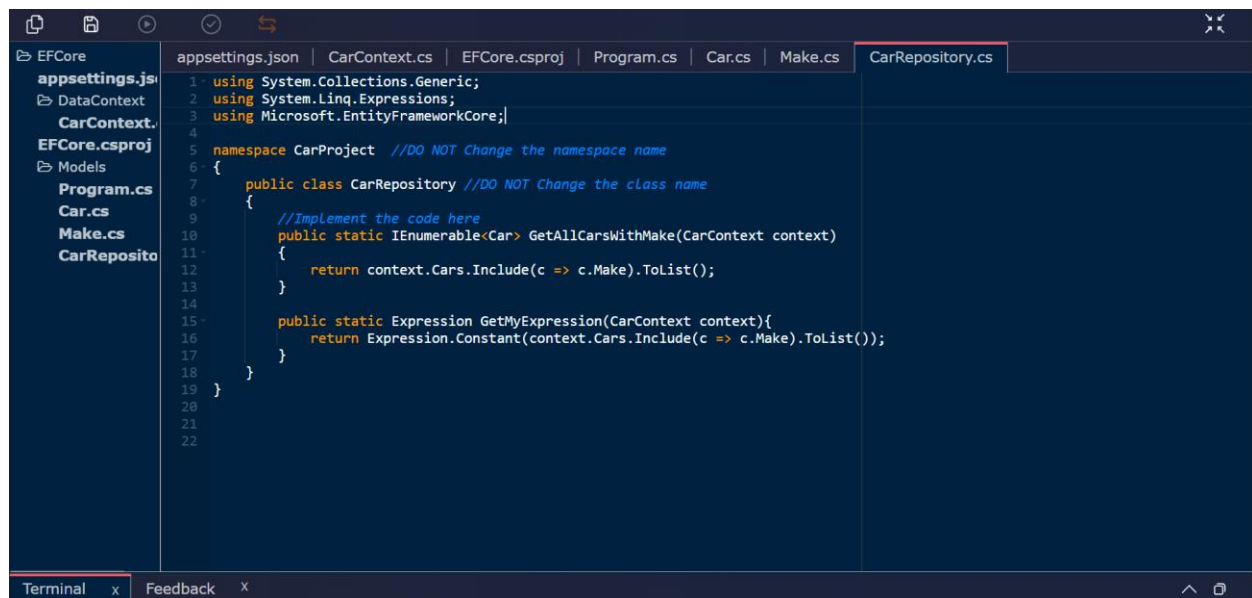
```
1 using System.ComponentModel.DataAnnotations;
2 using System.ComponentModel.DataAnnotations.Schema;
3
4 namespace CarProject //DO NOT Change the namespace name
5 {
6     public class Car //DO NOT Change the class name
7     {
8         //Implement the code here
9         [Key]
10         [DatabaseGenerated(DatabaseGeneratedOption.None)]
11         public int Id {get; set;}
12
13         public string Model {get; set;}
14         public int Year {get; set;}
15
16         public int MakeId {get; set;}
17         public Make Make {get; set;}
18     }
19 }
```



The screenshot shows a code editor with a sidebar on the left containing a file explorer. The file explorer shows a project named 'EFCore' with files: 'appsettings.json', 'DataContext', 'CarContext.cs', 'EFCore.csproj', 'Models', 'Program.cs', 'Car.cs', 'Make.cs', and 'CarRepository.cs'. The 'Make.cs' file is selected and open in the main editor. The code in 'Make.cs' is as follows:

```
1: using System.ComponentModel.DataAnnotations;
2: using System.ComponentModel.DataAnnotations.Schema;
3:
4: namespace CarProject //DO NOT Change the namespace name
5: {
6:     public class Make //DO NOT Change the class name
7:     {
8:         //Implement the code here
9:         [Key]
10:        [DatabaseGenerated(DatabaseGeneratedOption.None)]
11:        public int Id {get; set;}
12:
13:        public string Name {get; set;}
14:    }
15: }
```

At the bottom of the editor, there is a terminal window with the text: 'Terminal Feedback X' and a URL: 'nizant.tekstac.com/mod/vpl/forms/edit.php?id=22056&userid=20624#vpl\_file5'.



The screenshot shows the same code editor as above, but now the 'CarRepository.cs' file is selected and open in the main editor. The code in 'CarRepository.cs' is as follows:

```
1: using System.Collections.Generic;
2: using System.Linq.Expressions;
3: using Microsoft.EntityFrameworkCore;
4:
5: namespace CarProject //DO NOT Change the namespace name
6: {
7:     public class CarRepository //DO NOT Change the class name
8:     {
9:         //Implement the code here
10:        public static IEnumerable<Car> GetAllCarsWithMake(CarContext context)
11:        {
12:            return context.Cars.Include(c => c.Make).ToList();
13:        }
14:
15:        public static Expression GetMyExpression(CarContext context){
16:            return Expression.Constant(context.Cars.Include(c => c.Make).ToList());
17:        }
18:    }
19: }
```

At the bottom of the editor, there is a terminal window with the text: 'Terminal Feedback X'.



## Car Repository - Lazy Loading

---

### Description

#### Note :

**EFCore.csproj** file is given for your reference in the code template. Do not make any changes here.

**Objective:** This application aims to educate learners about the concept of lazy loading in the Entity Framework Core. Lazy Loading, the default behavior in Entity Framework Core, defers the loading of related entities until they are accessed for the first time.

#### Scenario:

Sam has been assigned the new task of creating an advanced application for retrieving car and make details from the company's database.

Help them create a **C#** application to retrieve details from the **SQLSERVER** database and use the **entity framework core** to connect to the database, which provides the following functionalities.

#### Functionalities:

- Retrieving car and make details from the database using **Lazy Loading**.

#### Requirements:

1. In the class **Car**, implement the below given public properties.

Data Type	Property Name
int	Id

string	Model
int	Year
int	MakeId
Make	Make

**Note :** To enable Lazy Loading, use the **virtual** keyword for **Make** Property.

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Car** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Car** class.

2. In the class **Make**, implement the below given public properties.

Data Type	Property Name
int	Id
string	Name

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Make** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Make** class.

3. In the class **Car Context**, implement the below given property, method and also inherit the class **DbContext**.

Data Type	Property Name
DbSet<Car>	Cars
DbSet<Make>	Makes

Method	Description
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)	This method is used to configure the database connection for the <b>DbContext</b> . <b>Note</b> : In this method, call the <b>UseLazyLoadingProxies()</b> method to enable lazy loading.

**appsettings.json:**

```
{  
  "ConnectionStrings": {  
    "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial  
Catalog=StudentDB;Integrated Security=True;"  
  }  
}
```

This file is already provided for you. Use the connection string in your context class **OnConfiguring** method to configure the database.

The below-given reference code is the part of the CarContext.cs

4. In the class **CarRepository**, implement the below given public method.

Method	Description
public static IEnumerable<Car> GetAllCarsWithMake(CarContext context)	This method is used to get the cars with make details from the database.

4. In the class **Program - Main()** method,

-- Call the GetAllCarsWithMake method and display the result as per the sample output.

5. The "**GetMyExpression**" method is for testing your LINQ QUERY EXPRESSION. So fill your query expression in the space holder provided. ONLY THE QUERY EXPRESSION  
Nothing more needs to be implemented in this method.

The below sample data is already available in the database table **Makes**.

Id	Name
----	------

101	Toyota
102	Ford
103	Honda
104	BMW

The below sample data is already available in the database table **Cars**.

Id	Model	Year	MakeId
1	Camry	1999	101
2	Mustang	2002	102
3	Sienna	2000	101
4	Civic	1998	103
5	X3	2002	104

**Note:**

- Keep the properties, methods and classes as **public**.
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

**Sample Output :**

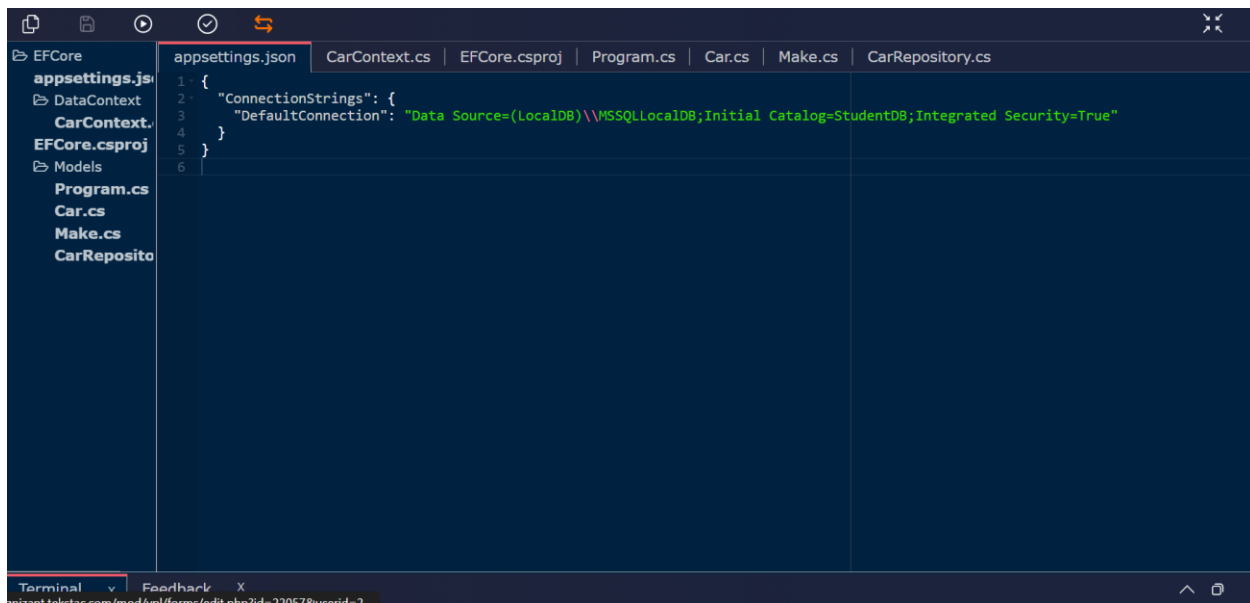
Car Id: 1,    Make: Toyota,    Model: Camry,    Year: 1999

Car Id: 2,    Make: Ford,        Model: Mustang,    Year: 2002

Car Id: 3,    Make: Toyota,     Model: Sienna,      Year: 2000

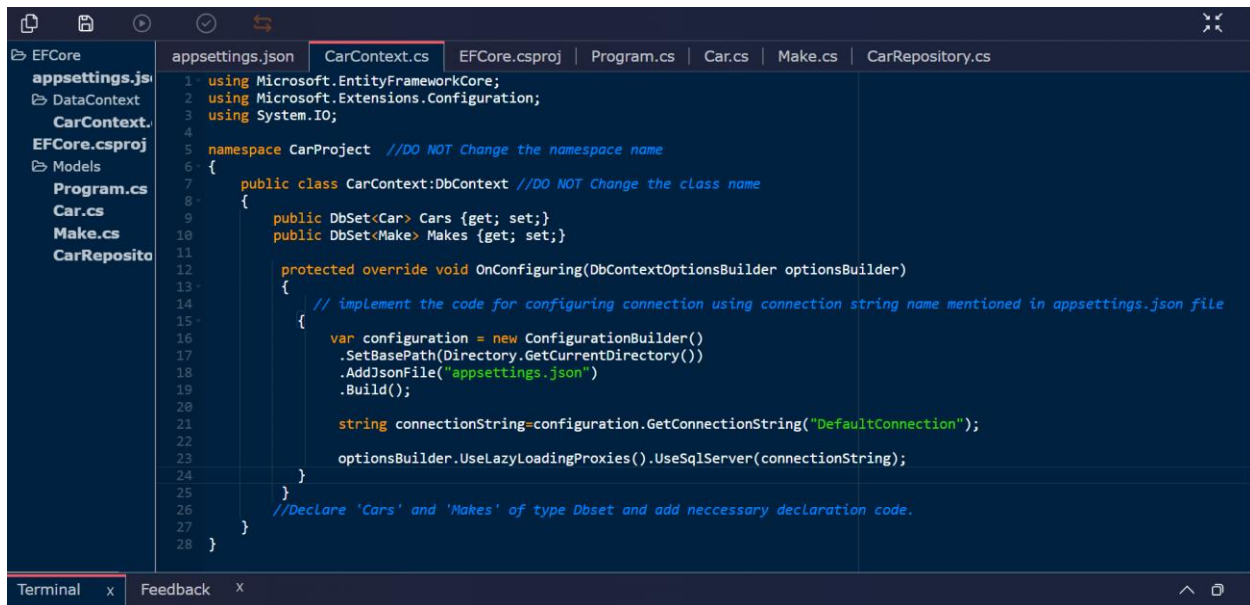
Car Id: 4,    Make: Honda,        Model: Civic,        Year: 1998

Car Id: 5,    Make: BMW,         Model: X3,           Year: 2002



The screenshot shows the Visual Studio Code editor with the EFCore project open. The file explorer on the left shows the project structure: appsettings.json, DataContext, CarContext, EFCore.csproj, Models, Program.cs, Car.cs, Make.cs, and CarRepository.cs. The appsettings.json file is selected and its content is displayed in the editor. The content is a JSON object with a 'ConnectionStrings' section containing a 'DefaultConnection' string.

```
1 {
2   "ConnectionStrings": {
3     "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True"
4   }
5 }
6
```



The screenshot shows the Visual Studio Code editor with the EFCore project open. The file explorer on the left shows the project structure: appsettings.json, DataContext, CarContext, EFCore.csproj, Models, Program.cs, Car.cs, Make.cs, and CarRepository.cs. The CarContext.cs file is selected and its content is displayed in the editor. The content is a C# class named CarContext:DbContext, which is a derived class of DbContext. It contains two DbSet properties: Cars and Makes. It also has an OnConfiguring method that configures the database connection using the connection string from appsettings.json.

```
1 using Microsoft.EntityFrameworkCore;
2 using Microsoft.Extensions.Configuration;
3 using System.IO;
4
5 namespace CarProject //DO NOT Change the namespace name
6 {
7   public class CarContext:DbContext //DO NOT Change the class name
8   {
9     public DbSet<Car> Cars {get; set;}
10    public DbSet<Make> Makes {get; set;}
11
12    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
13    {
14      // implement the code for configuring connection using connection string name mentioned in appsettings.json file
15      var configuration = new ConfigurationBuilder()
16        .SetBasePath(Directory.GetCurrentDirectory())
17        .AddJsonFile("appsettings.json")
18        .Build();
19
20      string connectionString=configuration.GetConnectionString("DefaultConnection");
21
22      optionsBuilder.UseLazyLoadingProxies().UseSqlServer(connectionString);
23    }
24  }
25  //Declare 'Cars' and 'Makes' of type DbSet and add neccessary declaration code.
26 }
27
28
```

This screenshot shows the Visual Studio IDE with the `EFCore.csproj` file open. The file is an XML project file for a .NET 6.0 project. It includes various package references for Entity Framework Core, Microsoft.Extensions.Configuration.Json, Microsoft.NET.Test.Sdk, NUnit, and NUnit3TestAdapter. The project is configured to generate a program file and to copy the `appsettings.json` file to the output directory.

```
<?xml version='1.0' encoding='utf-8'>
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <GenerateProgramFile>false</GenerateProgramFile>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Proxies" Version="8.0.0-preview.5.23280.1" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0-preview.5.23280.1" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0-preview.5.23280.1">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="8.0.0-preview.5.23280.8" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.7.0-preview.23280.1" />
    <PackageReference Include="NUnit" Version="3.13.3" />
    <PackageReference Include="NUnit3TestAdapter" Version="4.5.0" />
  </ItemGroup>
  <ItemGroup>
    <None Update="appsettings.json">
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </None>
  </ItemGroup>
</Project>
```

This screenshot shows the Visual Studio IDE with the `Program.cs` file open. The file contains the main entry point of the application. It includes using statements for System, System.Collections.Generic, System.Linq.Expressions, and Microsoft.EntityFrameworkCore. The `Program` class has a `Main` method that creates a `CarContext` instance, retrieves all cars with their make information from the `CarRepository`, and prints them to the console. It also includes a sample LINQ expression for testing purposes.

```
using System;
using System.Collections.Generic;
using System.Linq.Expressions;
using Microsoft.EntityFrameworkCore;

namespace CarProject //DO NOT Change the namespace name
{
    public class Program //DO NOT Change the class name
    {
        public static void Main(string[] args)
        {
            //Implement the code here
            using (var context=new CarContext())
            {
                var cars = CarRepository.GetAllCarsWithMake(context);
                foreach (var car in cars)
                {
                    Console.WriteLine($"Car Id:{car.Id}, Make:{car.Make.Name}, Model:{car.Model}, Year:{car.Year}");
                }
            }

            public static ParameterExpression variableExpr = Expression.Variable(typeof(IEnumerable<Car>), "sampleVar");
            public static Expression GetMyExpression(CarContext context)
            {
                Expression assignExpr = Expression.Assign(variableExpr, Expression.Constant(CarRepository.GetAllCarsWithMake(context)));
                return assignExpr;
            }
        }
    }
}
```

```
1 using System.ComponentModel.DataAnnotations;
2 using System.ComponentModel.DataAnnotations.Schema;
3
4 namespace CarProject //DO NOT Change the namespace name
5 {
6     public class Car //DO NOT Change the class name
7     {
8         //Implement the code here
9         [Key]
10        [DatabaseGenerated(DatabaseGeneratedOption.None)]
11
12        public int Id {get; set;}
13        public string Model {get; set;}
14        public int Year {get; set;}
15        public int MakeId {get; set;}
16
17        public virtual Make Make {get; set;}
18    }
19 }
```

```
1 using System.ComponentModel.DataAnnotations;
2 using System.ComponentModel.DataAnnotations.Schema;
3
4 namespace CarProject //DO NOT Change the namespace name
5 {
6     public class Make //DO NOT Change the class name
7     {
8         //Implement the code here
9         [Key]
10        [DatabaseGenerated(DatabaseGeneratedOption.None)]
11        public int Id{get; set;}
12
13        public string Name {get; set;}
14        // public virtual ICollection<Car> Cars {get; set;}
15    }
16 }
```



