Add EF Core to your project

Using the NuGet package manager or NuGet package manager console, you can create a connection between your project and the SQL Server.

NuGet package manager (Windows only)

In your solution explorer, right click your project. (Not the solution) Then click manage NuGet packages. Click browse on the new window that appeared and search for the following. Click install on each.

- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools

You can confirm this is installed by clicking the installed tab. You should see both here. If you don't make sure you do not have a filter set.

Console (Mac)

Before you can use any of the commands, you must first install the dotnet commands. The following command will install it globally, so you only need to run this command once and you should be good to go.

Tools > NuGet Package Manager > Package Manager Console

```
dotnet tool install --global dotnet-ef
```

Once installed, you can run the following commands on any project to add Entity Framework.

```
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
```

```
dotnet add package Microsoft.EntityFrameworkCore
```

```
dotnet add package Microsoft.EntityFrameworkCore.Tools
```

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer
Install-Package Microsoft.EntityFrameworkCore
Install-Package Microsoft.EntityFrameworkCore.Tools
```

Build your models

There are two techniques:

- With **database-first**, you create the tables in SQL then have EF Core create corresponding C# classes for you.
- With **code-first**, you define the C# classes, and EF Core creates the database tables for you.

Database First

This will generate C# classes based on a database.

Start by creating a new Database on your SQL Server. Then create a new table(s). From there you will take the following command to the package manager console.

Tools > NuGet Package Manager > Package Manager Console

Windows local server

```
Scaffold-DbContext 'Data Source=.\sqlexpress;Initial Catalog=CarDB;
Integrated Security=SSPI;Encrypt=false;TrustServerCertificate=True;'
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

This command may need to be adjusted. Here are the common ones to change

- Data Source - This is your *server name*.. You can find the server name in SSMS:
  - Right click on the server at the top of "Object Explorer"
  - then click Properties → "view connection properties" (near lower left corner)
  - the Server Name is listed in the middle section called "Product"
- .\sqlexpress is the most common one but if your local server has a different name this is the one you adjust.
- Initial Catalog - This is your database name. Adjust this every time you run the command to your server
- -OutputDir
  - This is the folder where your files will be stored. No need to change this unless you want a different folder name.

Mac local server or Azure Server

```
dotnet ef dbcontext scaffold 'Server=localhost,1433; Initial
Catalog=CarDB; User ID=SA; Password=EnterPasswordHere1;
TrustServerCertificate=true;' Microsoft.EntityFrameworkCore.SqlServer
-o Models
```

This command may need to be adjusted. Here are the common ones to change

- Server
    - This is your server name.
        - Mac: localhost is the most common one but if your local server has a different name this is the one you adjust. The , after is your port. 1433 is the most common one.
        - Azure: Azure will provide you with a server to plug in.
- Initial Catalog
    - This is your database name. Adjust this every time you run the command to your server
- User ID
    - This is your login user name. You should have this from when you created your Server.
- Password
    - This is your login password. You should have this from when you created your Server.
- -OutputDir
    - This is the folder where your files will be stored. No need to change this unless you want a different folder name.

Code First

This will generate a database based on C# classes.

Models

Create the classes you will need for your project. Each class maps to a SQL table. To create a column for your database, create a property.

```
public string Name {get; set;}
```

To mark one as a Primary key, add the following.

```
[Key]
public int Id {get; set;}
```

You will also need the following using statement.

```
using System.ComponentModel.DataAnnotations;
```

DbContext

Create a class. This class should be named based on what you want to call the database followed by Context. Example: CarContext

This new class should be a child of DbContext. You will need to add the following using statement.

```
using Microsoft.EntityFrameworkCore;
```

Add a DbSet property for each table you want in your database.

```
public DbSet<Car> Cars { get; set; }
```

Override the OnConfiguring method.

- Replace the Server to match your server.
- Replace the database with what you would like the database to be named.
- If you are on Mac or using an Azure Server,
    - Replace: Integrated Security=SSPI;
    - With: User ID=SA; Password=EnterPasswordHere1;
        - Replace values with your user id and password

```
protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder){
optionsBuilder.UseSqlServer(
@"Server=.\SQLEXPRESS;Database=efmvc1;Integrated Security=SSPI;");
}
```

Tools > NuGet Package Manager > Package Manager Console

On the following command, replace CreateCarDB with whatever you would like to call this. I recommend CreateDatabaseNameDB. Then run it.

```
add-migration CreateCarDB
```

```
Update-database
```

Wait for it all to run, then you should be good.

Additional resources

- Getting Started - EF Core | Microsoft Learn - Example of the Code First approach.