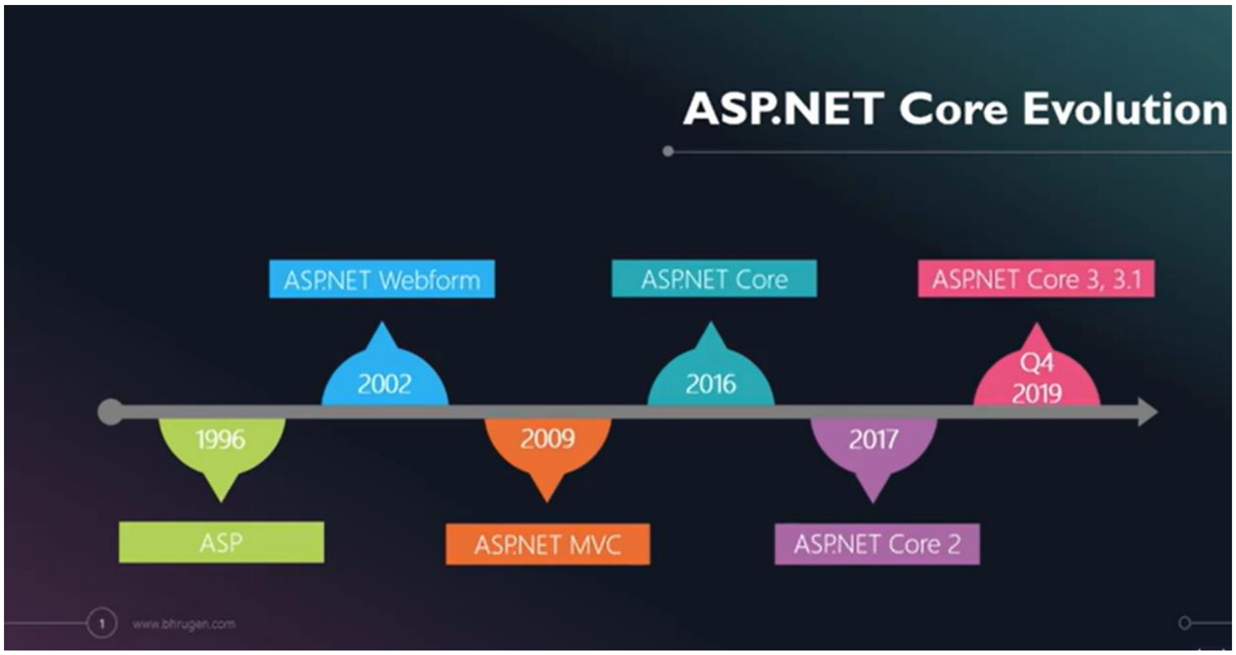
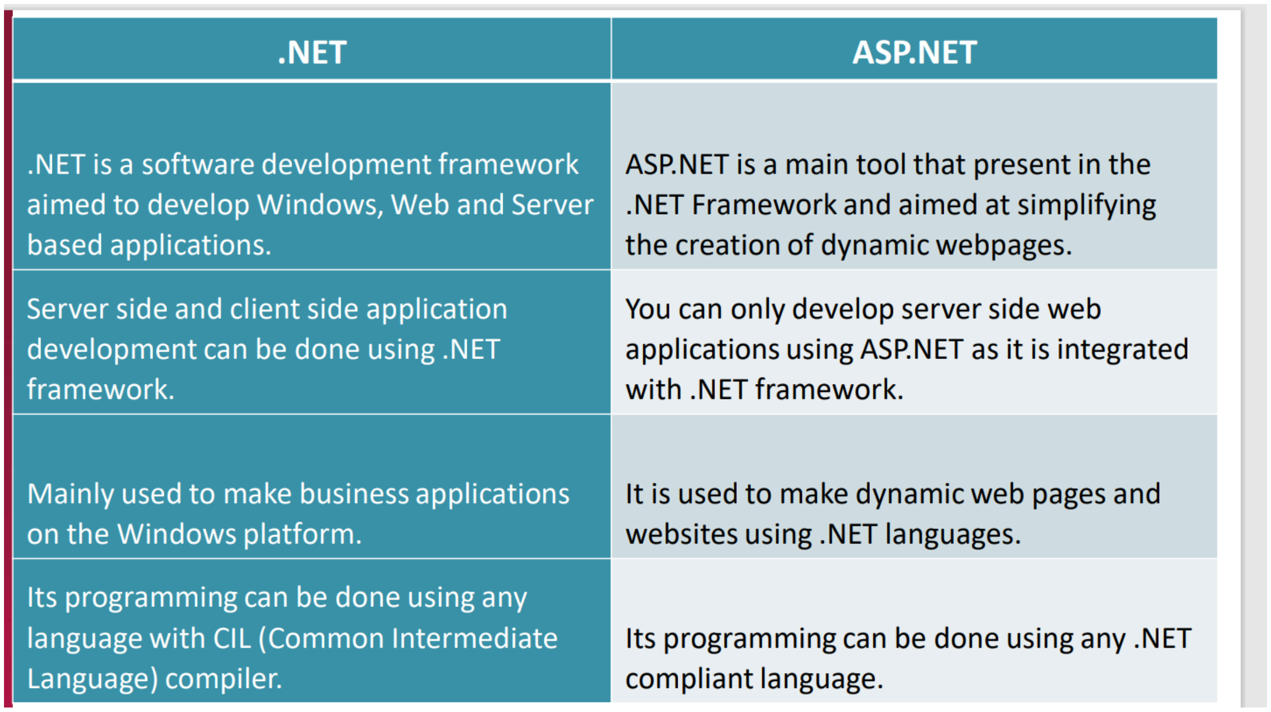
Unit 2: Introduction to ASP.NET

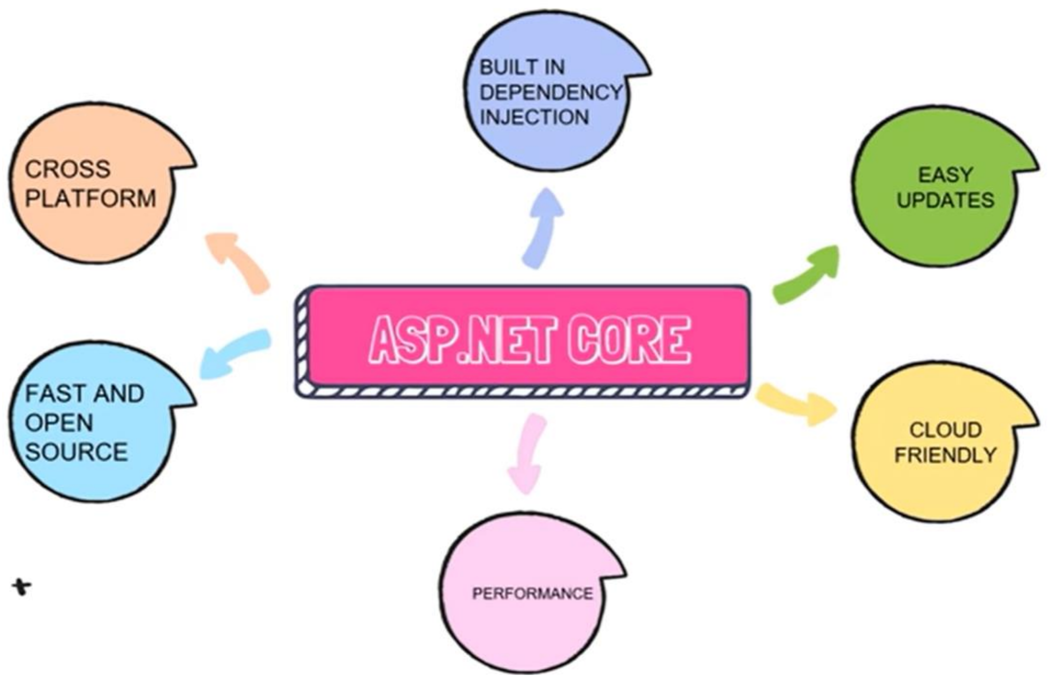
Microsoft ASP.NET Core is part of a history of Microsoft technologies used to build websites and web services.

- ASP.NET is a web application framework designed and developed by Microsoft.
- a subset of the .NET Framework and successor of the classic ASP (Active Server Pages).
- With version 1.0 of the .NET Framework, it was first released in January 2002.
- before the year 2002 for developing web applications and services, there was Classic ASP.





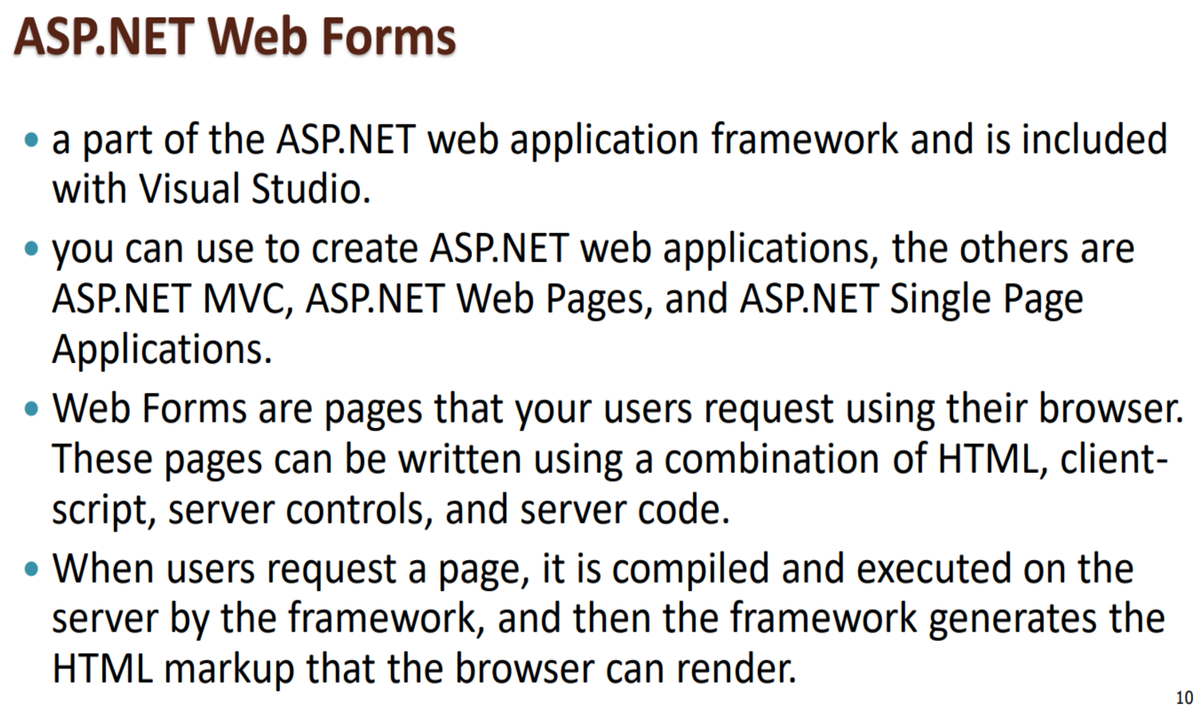
.NET	ASP.NET
.NET is a software development framework aimed to develop Windows, Web and Server based applications.	ASP.NET is a main tool that present in the .NET Framework and aimed at simplifying the creation of dynamic webpages.
Server side and client side application development can be done using .NET framework.	You can only develop server side web applications using ASP.NET as it is integrated with .NET framework.
Mainly used to make business applications on the Windows platform.	It is used to make dynamic web pages and websites using .NET languages.
Its programming can be done using any language with CIL (Common Intermediate Language) compiler.	Its programming can be done using any .NET compliant language.



Classic ASP.NET versus modern ASP.NET Core

Until now, ASP.NET has been built on top of a large assembly in the .NET Framework named System.Web.dll and it is tightly coupled to Microsoft's Windows-only web server named Internet Information Services (IIS). Over the years, this assembly has accumulated a lot of features, many of which are not suitable for modern cross-platform development.

ASP.NET Core is a major redesign of ASP.NET. It removes the dependency on the System.Web.dll assembly and IIS and is composed of modular lightweight packages, just like the rest of .NET Core. We can develop and run ASP.NET Core applications cross-platform on Windows, macOS, and Linux. Microsoft has even created a cross-platform, super-performant web server named Kestrel, and the entire stack is open source.



ASP.NET Web Forms

- a part of the ASP.NET web application framework and is included with Visual Studio.
- you can use to create ASP.NET web applications, the others are ASP.NET MVC, ASP.NET Web Pages, and ASP.NET Single Page Applications.
- Web Forms are pages that your users request using their browser. These pages can be written using a combination of HTML, client-script, server controls, and server code.
- When users request a page, it is compiled and executed on the server by the framework, and then the framework generates the HTML markup that the browser can render.

- An ASP.NET Web Forms page presents information to the user in any browser or client device.
- The Visual Studio (IDE) lets you drag and drop server controls to lay out your Web Forms page. You can then easily set properties, methods, and events for controls on the page or for the page itself. These properties, methods, and events are used to define the web page's behavior, look and feel, and so on
- Based on Microsoft ASP.NET technology, in which code that runs on the server dynamically generates Web page output to the browser or client device.

Features of ASP.NET Web Forms

- **Server Controls-** ASP.NET Web server controls are similar to familiar HTML elements, such as buttons and text boxes. Other controls are calendar controls, and controls that you can use to connect to data sources and display data.
- **Master Pages-** ASP.NET master pages allow you to create a consistent layout for the pages in your application. A single master page defines the look and feel and standard behavior for all of the pages (or a group of pages) in your application. You can then create individual content pages along with the master page to render the web page.
- **Working with Data**- ASP.NET provides many options for storing, retrieving, and displaying data in web page UI elements such as tables and text boxes and drop-down lists.

- **Client Script and Client Frameworks -** You can write client-script functionality in ASP.NET Web Form pages to provide responsive user interface to users. You can also use client script to make asynchronous calls to the Web server while a page is running in the browser.
- **Routing -** URL routing allows you to configure an application to accept request URL. A request URL is simply the URL a user enters into their browser to find a page on your web site. You use routing to define URLs that are semantically meaningful to users and that can help with search-engine optimization (SEO).
- **State Management -** ASP.NET Web Forms includes several options that help you preserve data on both a per-page basis and an application-wide basis.
- **Security -** offer features to develop secure application from various security threats.

- **Performance –** offers performance related to page and server control processing, state management, data access, application configuration and loading, and efficient coding practices.
- **Internationalization -** enables you to create web pages that can obtain content and other data based on the preferred language setting or localized resource for the browser or based on the user's explicit choice of language. Content and other data is referred to as resources and such data can be stored in resource files or other sources.
- **Debugging and Error Handling -** diagnose problems that might arise in application. Debugging and error handling are well so that applications compile and run effectively.
- **Deployment and Hosting-** Visual Studio, ASP.NET, Azure, and IIS provide tools that help you with the process of deploying and hosting your application

Working with Databases Using Entity Framework Core

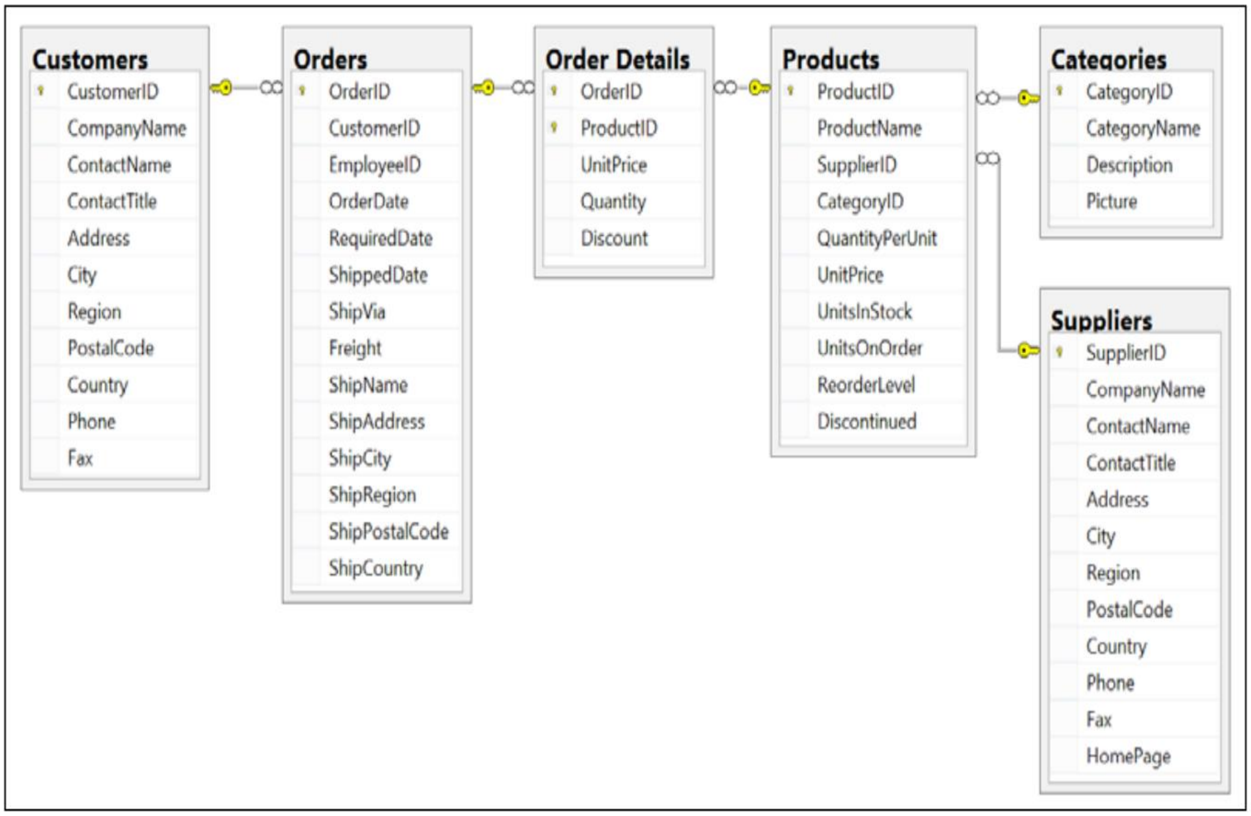
Two of the most common places to store data are in a Relational Database Management System (RDBMS) such as Microsoft SQL Server, PostgreSQL, MySQL, and SQLite, or in a NoSQL data store such as Microsoft Azure Cosmos DB, Redis, MongoDB, and Apache Cassandra.

Understanding Entity Framework

Entity Framework (EF) was first released as part of the .NET Framework 3.5 with Service Pack 1 back in late 2008. Since then, Entity Framework has evolved, as Microsoft has observed how programmers use an object-relational mapping (ORM)

tool in the real world. ORMs use a mapping definition to associate columns in tables to properties in classes. Then, a programmer can interact with objects of different types in a way that they are familiar with, instead of having to deal with knowing how to store the values in a relational table or other structure in the database.

A Sample Database, Northwind



We will be using sqlite dbms:

1. Goto: https://www.sqlite.org/download.html and download zip file, sqlite-tools-win32-x86-3410200.zip from **Precompiled Binaries for Windows.**
2. **Extract all from the zip file to a folder c:\sqlite, and add this path to the environment variable in Windows system.**
3. **Now create a folder for database placing, I have done it on c:\database**
4. Download the Northwind.sql script from github and run the following command.

```
C:\Database\sqlite3 Northwind.db < Northwind.sql.txt
```

This will create our Northwind database.

5. Download and install Sqlite Studio from https://sqlitestudio.pl/
6. In Sqlitestudio, click on Database menu and add your database by locating the Northwind.db in the folder (mine is c:\Database).

Now, you can have detail look of structure of tables and the data placed in each table easily.

Choosing an EF Core data provider

To manage data in a specific database, we need classes that know how to efficiently talk to that database. EF Core data providers are sets of classes that are optimized for a specific data store. There is even a provider for storing the data in the memory of the current process, which is useful for high-performance unit testing since it avoids hitting an external system.

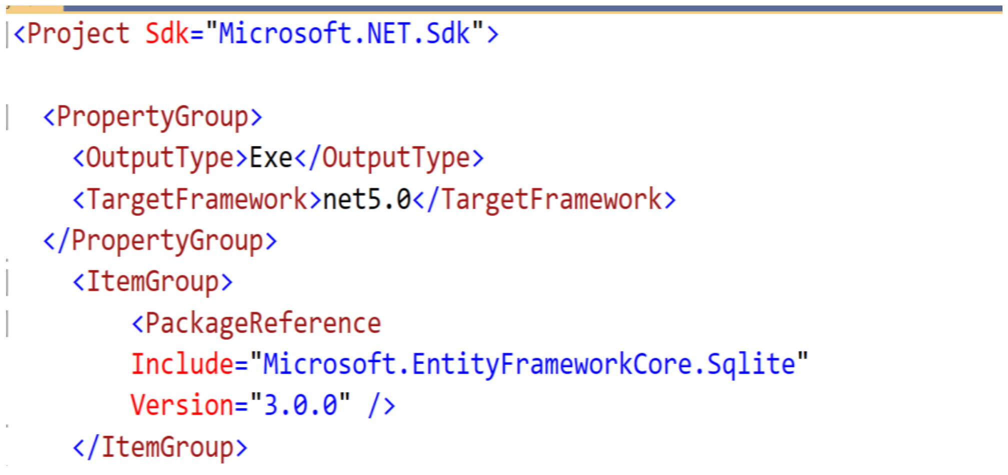
They are distributed as NuGet packages, as shown in the following table:

To manage this data store	Install this NuGet package
Microsoft SQL Server 2008 or later	`Microsoft.EntityFrameworkCore.SqlServer`
SQLite 3.7 or later	`Microsoft.EntityFrameworkCore.SQLite`
MySQL	`MySQL.Data.EntityFrameworkCore`
In-memory	`Microsoft.EntityFrameworkCore.InMemory`

More Information: Devart is a third party that offers EF Core providers for a wide range of data stores. Find out more at the following link: `https://www.devart.com/dotconnect/entityframework.html`

Connecting with database:

1. Now, go to VS and Create a new console App in the folder where you have your Northwind database.
2. Edit the database.csproj file as below:



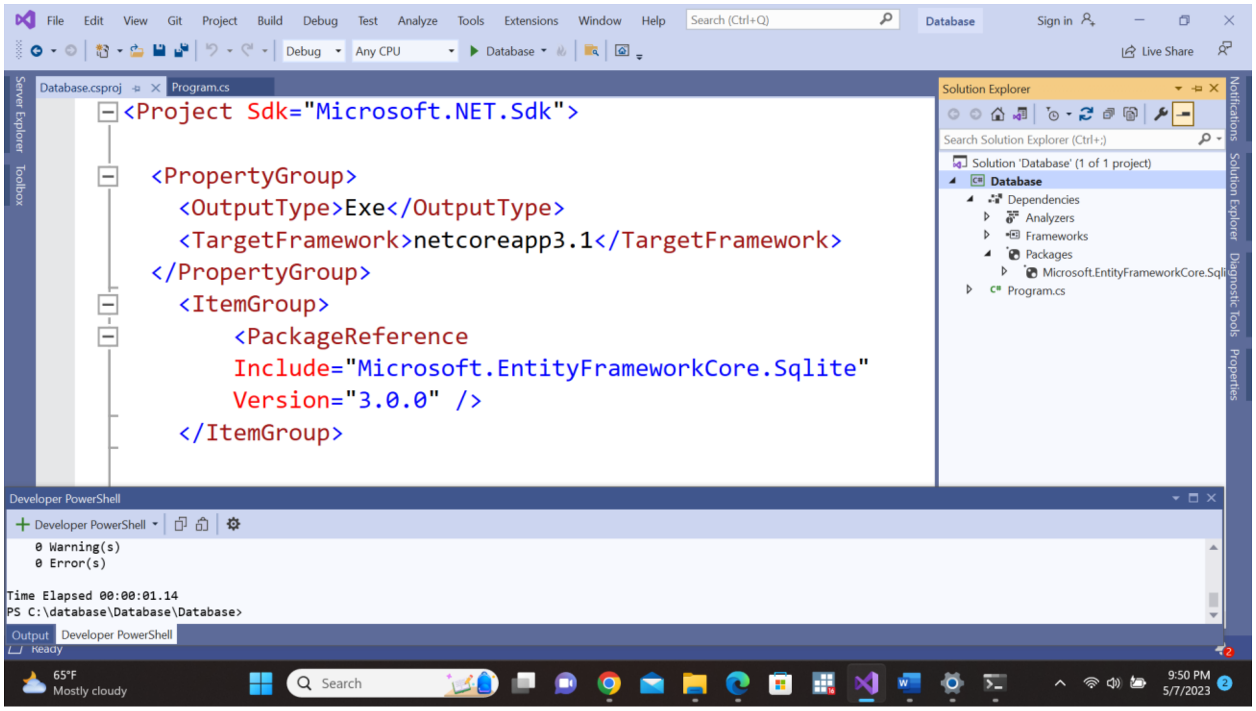
```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>
    <ItemGroup>
        <PackageReference
        Include="Microsoft.EntityFrameworkCore.Sqlite"
        Version="3.0.0" />
    </ItemGroup>
```

```
<ItemGroup>
    <PackageReference
   Include="Microsoft.EntityFrameworkCore.Sqlite"
   Version="3.0.0" />
 </ItemGroup>
```

3. Then right click on the project name and open the terminal.
4. Then run the build command on Terminal: *dotnet build*.

You can see the packages added in our project:



```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>
    <ItemGroup>
        <PackageReference
        Include="Microsoft.EntityFrameworkCore.Sqlite"
        Version="3.0.0" />
    </ItemGroup>
```

```
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:01.14
PS C:\database\Database\Database>
```

Now, we add three .cs classes for referring to two tables and one for database. See the codes for Northwind.cs, Product.cs and Category.cs, available on github under NorthwindDB folder.