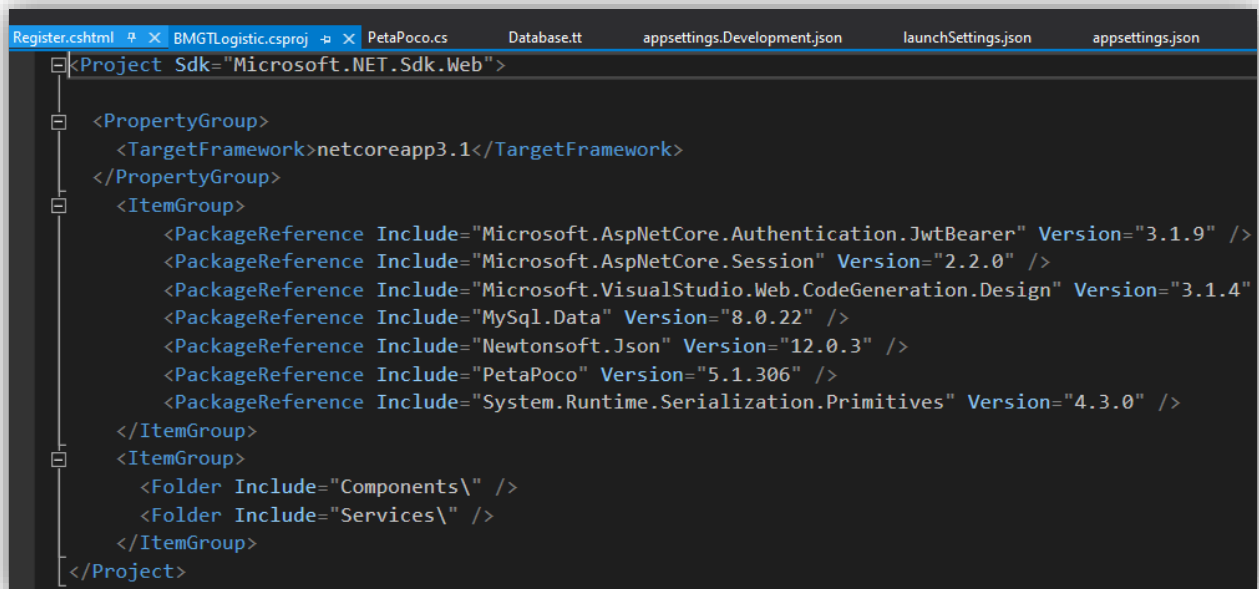


Petapoco Connectivity with DotNet Core

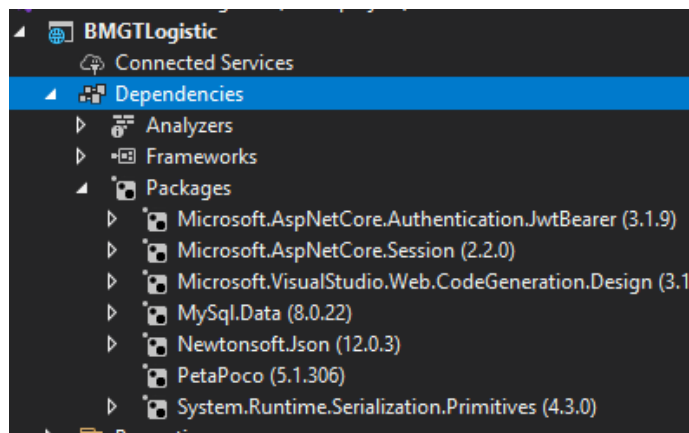
Double click on Project or Open project .csproj file Copy paste the following packages



```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="3.1.9" />
    <PackageReference Include="Microsoft.AspNetCore.Session" Version="2.2.0" />
    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="3.1.4" />
    <PackageReference Include="MySQL.Data" Version="8.0.22" />
    <PackageReference Include="Newtonsoft.Json" Version="12.0.3" />
    <PackageReference Include="PetaPoco" Version="5.1.306" />
    <PackageReference Include="System.Runtime.Serialization.Primitives" Version="4.3.0" />
  </ItemGroup>
  <ItemGroup>
    <Folder Include="Components\" />
    <Folder Include="Services\" />
  </ItemGroup>
</Project>
```

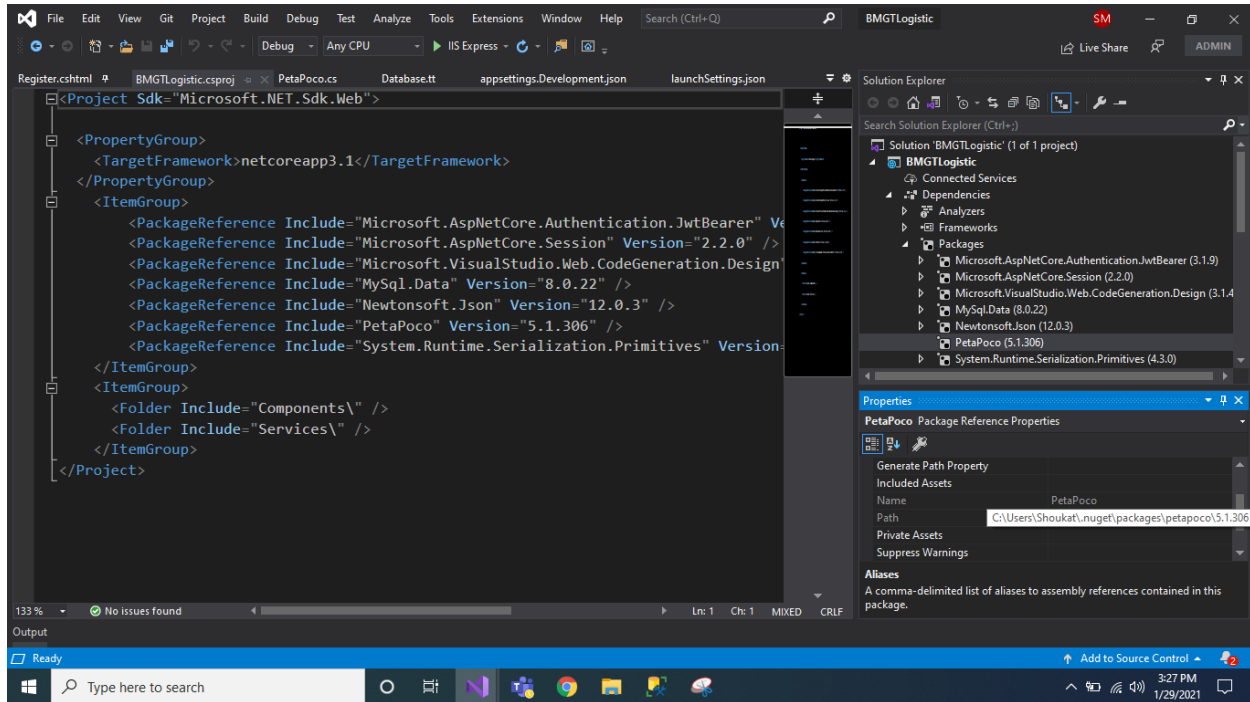
Press cntrl S to save configuration .Now to check that the following packages is added or not

```
<ItemGroup>
<PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="3.1.9" />
<PackageReference Include="Microsoft.AspNetCore.Session" Version="2.2.0" />
<PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="3.1.4" />
<PackageReference Include="MySQL.Data" Version="8.0.22" />
<PackageReference Include="Newtonsoft.Json" Version="12.0.3" />
<PackageReference Include="PetaPoco" Version="5.1.306" />
<PackageReference Include="System.Runtime.Serialization.Primitives" Version="4.3.0" />
</ItemGroup>
```

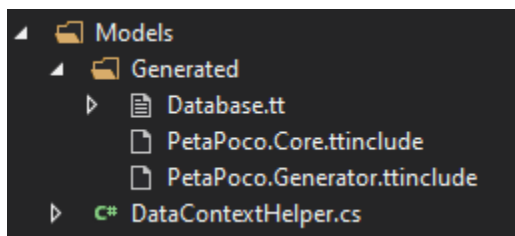


As we can see that all our required packages are installed successfully

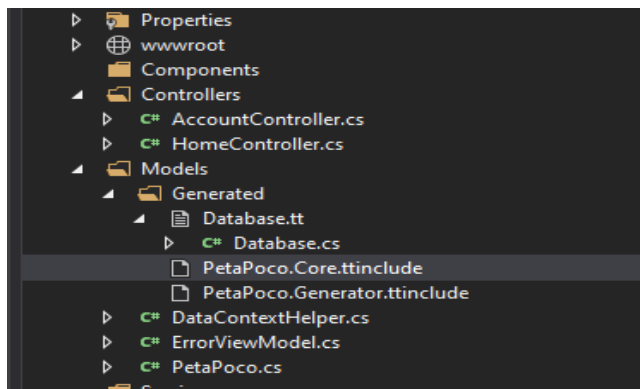
Now right click on Peta Poco Package and go to properties



Open this location and cut the Peta Poco Configuration and now copy past in Model folder



Now open the .ttinclude file

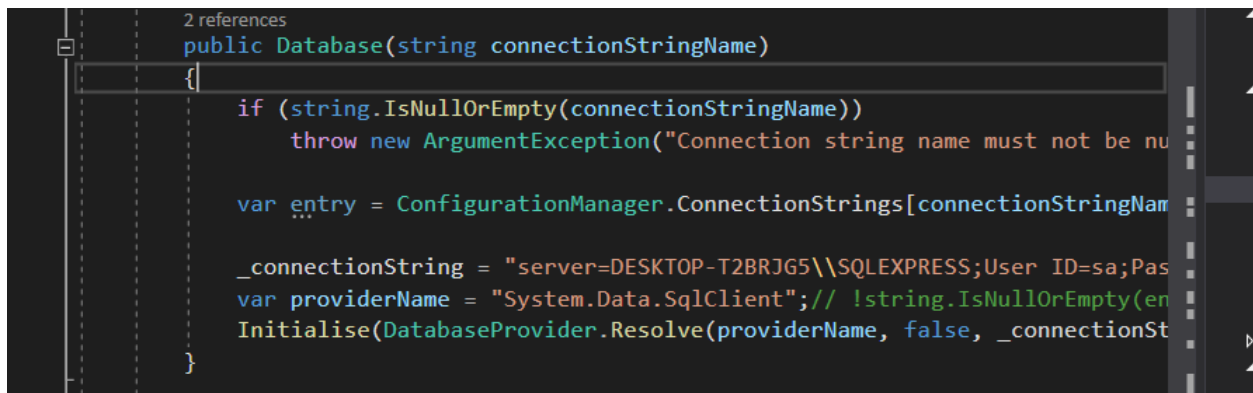


Open this file and Search this method **GetConnectionString()**

Copy paste this code.

```
connectionStringName = "FreightMasterConnection";
string result="server=DANATFP-SHOUKAT\\SQLEXPRESS;User
ID=sa;Password=123456;database=FreightMaster;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSu
bnetFailover=False;";
providerName="System.Data.SqlClient";
return result;
```

Now open the PetaPoco.cs file and search the this method

A screenshot of a code editor with a dark theme. It shows a C# method named 'Database' that takes a 'connectionStringName' parameter. The method first checks if the parameter is null or empty and throws an 'ArgumentException' if so. Then, it retrieves the connection string from 'ConfigurationManager.ConnectionStrings'. Finally, it initializes a 'DatabaseProvider' with the retrieved connection string and the provider name 'System.Data.SqlClient'.

```
2 references
public Database(string connectionStringName)
{
    if (string.IsNullOrEmpty(connectionStringName))
        throw new ArgumentException("Connection string name must not be nu

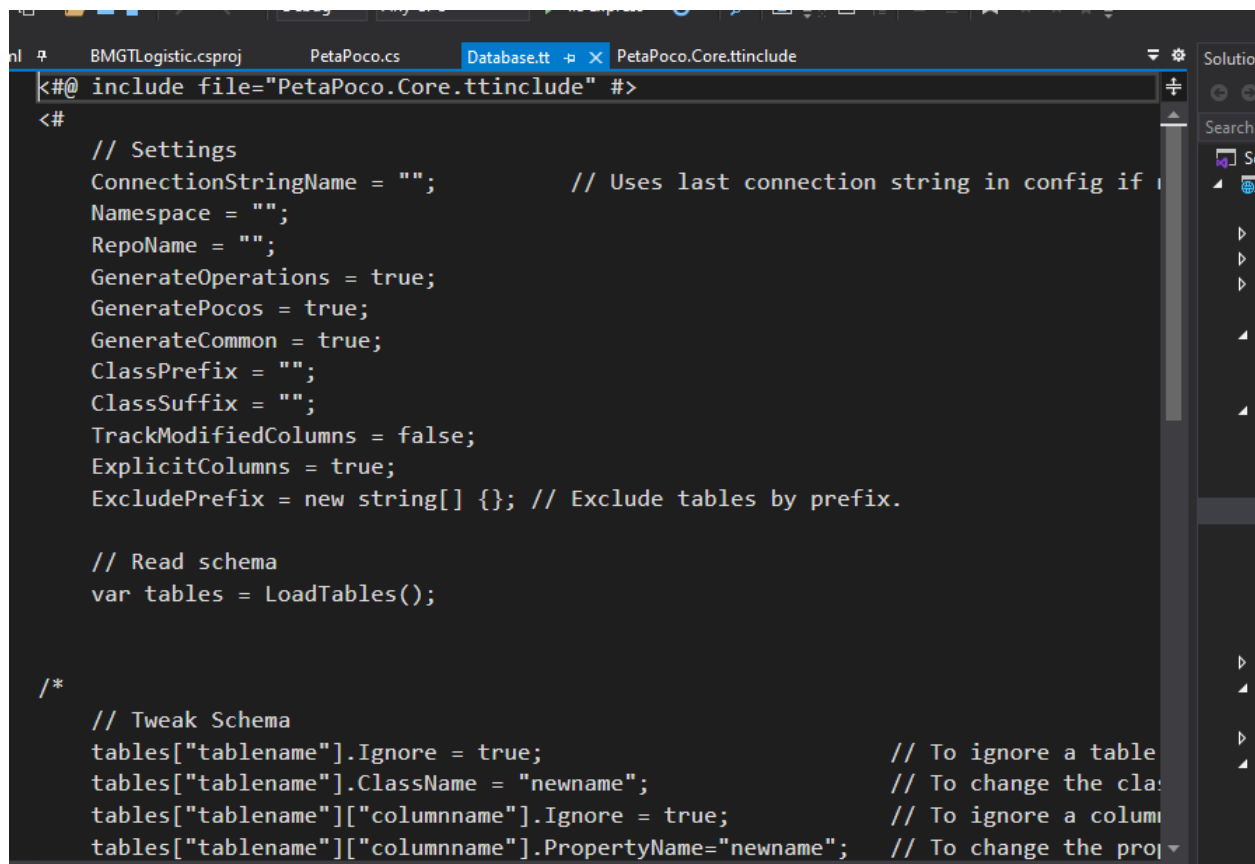
    var entry = ConfigurationManager.ConnectionStrings[connectionStringNam

    _connectionString = "server=DESKTOP-T2BRJG5\\SQLEXPRESS;User ID=sa;Pas
    var providerName = "System.Data.SqlClient"; // !string.IsNullOrEmpty(en
    Initialise(DatabaseProvider.Resolve(providerName, false, _connectionSt
}
```

Commit the above method and put this code below from the above method.

```
_connectionString = "server=DANATFP-SHOUKAT\\SQLEXPRESS;User
ID=sa;Password=123456;database=FreightMaster;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSu
bnetFailover=False;";
var providerName = "System.Data.SqlClient";
```

Now open this file and press Ctrl Save button to generate the peta poco of the database.



```
<#@ include file="PetaPoco.Core.ttinclude" #>
<#
    // Settings
    ConnectionStringName = "";           // Uses last connection string in config if
    Namespace = "";
    RepoName = "";
    GenerateOperations = true;
    GeneratePocos = true;
    GenerateCommon = true;
    ClassPrefix = "";
    ClassSuffix = "";
    TrackModifiedColumns = false;
    ExplicitColumns = true;
    ExcludePrefix = new string[] {}; // Exclude tables by prefix.

    // Read schema
    var tables = LoadTables();

    /*
    // Tweak Schema
    tables["tablename"].Ignore = true;           // To ignore a table
    tables["tablename"].ClassName = "newname";   // To change the class name
    tables["tablename"]["columnname"].Ignore = true; // To ignore a column
    tables["tablename"]["columnname"].PropertyName="newname"; // To change the property name
    */
}
```

Topic 2

Data Context Class

```

6
7 namespace newCoreApp.Models
8
9 public class DataContextHelper
10 {
11     1 reference
12     private static BMGTConnectionDB GetNewDataContext(string connectionStringName, bool enableAutoSelect
13     {
14         BMGTConnectionDB repository = new BMGTConnectionDB(connectionStringName);
15         repository.EnableAutoSelect = enableAutoSelect;
16         return (repository);
17     }
18
19     0 references
20     public static BMGTConnectionDB GetCPDataContext(bool enableAutoSelect = true)
21     {
22         return (GetNewDataContext("DefaultConnection", enableAutoSelect));
23     }
24 }

```

Create a datacontextHelper class and paste the following code

```

private static BMGTConnectionDB GetNewDataContext(string connectionStringName, bool
enableAutoSelect)
{
    BMGTConnectionDB repository = new BMGTConnectionDB(connectionStringName);
    repository.EnableAutoSelect = enableAutoSelect;
    return (repository);
}

public static BMGTConnectionDB GetCPDataContext(bool enableAutoSelect = true)
{
    return (GetNewDataContext("DefaultConnection", enableAutoSelect));
}

```

Adding custom routes

```
endpoints.MapControllerRoute(  
    name: "login",  
    pattern: "/login",  
    defaults: new { controller = "Authentication", action = "Login" }  
);
```

```
app.UseEndpoints(endpoints =>  
{  
    endpoints.MapControllerRoute(  
        name: "login",  
        pattern: "/login",  
        defaults: new { controller = "Account", action = "Login" }  
    );  
    endpoints.MapControllerRoute(  
        name: "default",  
        pattern: "{controller=Home}/{action=Index}/{id?}");  
});  
}
```