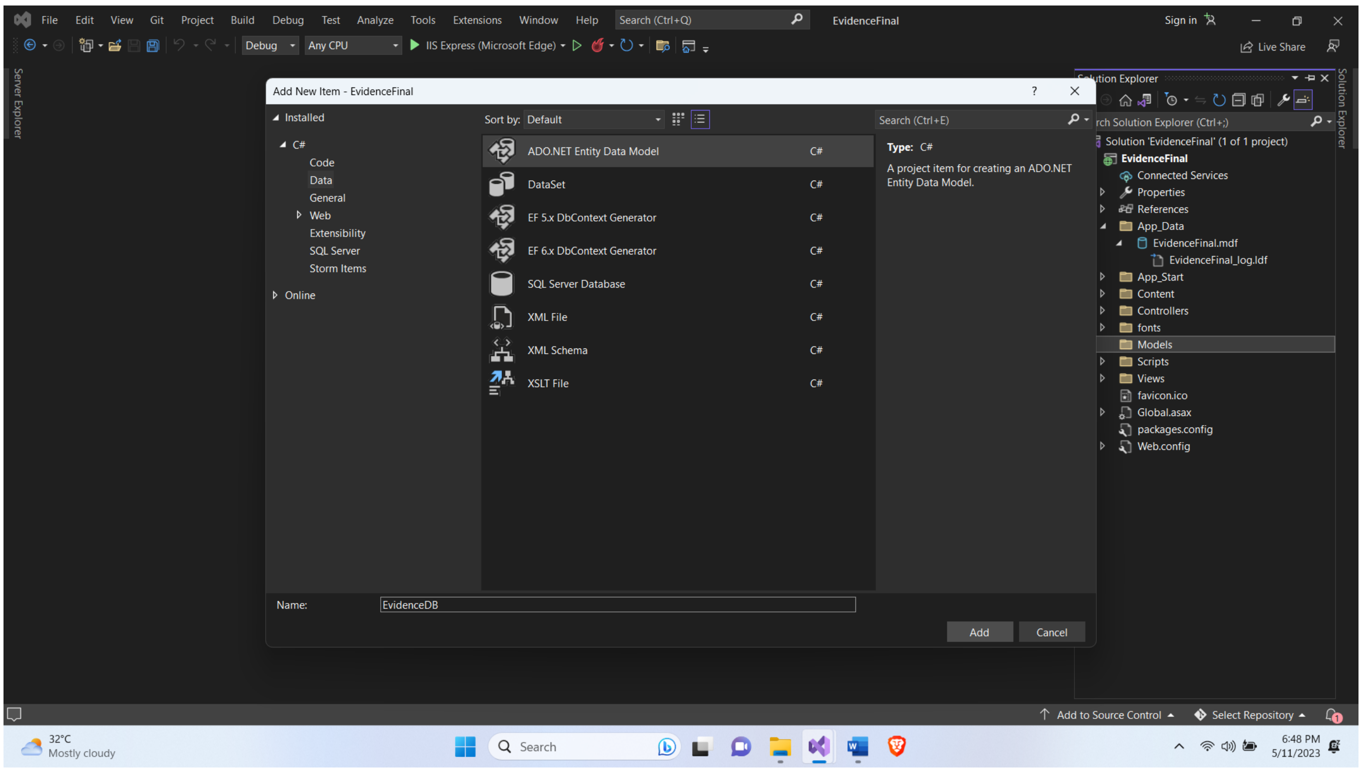
1. Create a project in **Asp.net web application (.net framework)** takes template **MVC.**
2. Add a database in **App data** folder with three relational table '**Customers'** '**Items'** and '**Orders'**

   Here are three table scripts:

```
CREATE TABLE [dbo].[Customers] (
[CustomerID]      INT           NOT NULL,
[CutomerName]     VARCHAR (50)  NULL,
[BillingAddress]  VARCHAR (50)  NULL,
[Imagepath]       VARCHAR (MAX) NULL,
PRIMARY KEY CLUSTERED ([CustomerID] ASC)
  );


CREATE TABLE [dbo].[items] (
[ItemID]   INT          IDENTITY (1, 1) NOT NULL,
[ItemName] VARCHAR (30) NULL,
PRIMARY KEY CLUSTERED ([ItemID] ASC)
);


CREATE TABLE [dbo].[Orders] (
[OrderID]     INT      NOT NULL,
[OrderNo]     INT      NOT NULL,
[CustomerID]  INT      NULL,
[ItemID]      INT      NULL,
[orderDate]   DATETIME NULL,
[OrderStatus] BIT      NULL,
PRIMARY KEY CLUSTERED ([OrderID] ASC),CONSTRAINT [FK_dbo.Orders_dbo.Customers_CustomerID]
FOREIGN KEY ([CustomerID]) REFERENCES      [dbo].[Customers] ([CustomerID]),
CONSTRAINT [FK_Orders_dbo.Items_ItemID] FOREIGN KEY ([ItemID]) REFERENCES [dbo].[items]
([ItemID])
);
```

3. Add **Ado.Net Entity Data Model** in **Model** Folder to add your database models. Rebuild project after entity model added. Here are some screenshot for better understand –



Add New Item - EvidenceFinal

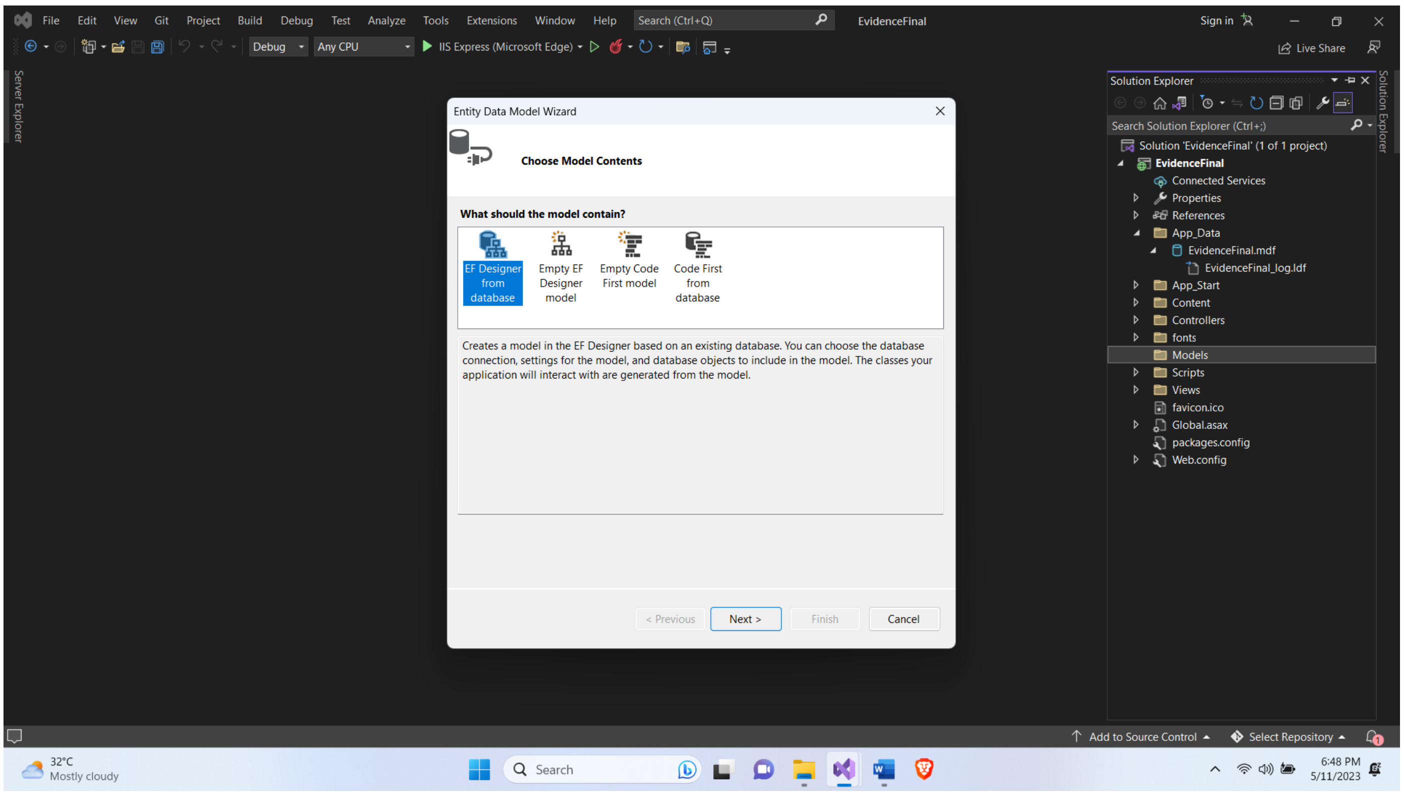
ADO.NET Entity Data Model	C#
DataSet	C#
EF 5.x DbContext Generator	C#
EF 6.x DbContext Generator	C#
SQL Server Database	C#
XML File	C#
XML Schema	C#
XSLT File	C#

Type: C#

A project item for creating an ADO.NET Entity Data Model.

Name: EvidenceDB

Solution 'EvidenceFinal' (1 of 1 project)
- **EvidenceFinal**
  - Connected Services
  - Properties
  - References
  - App_Data
    - EvidenceFinal.mdf
      - EvidenceFinal_log.ldf
  - App_Start
  - Content
  - Controllers
  - fonts
  - Models
  - Scripts
  - Views
  - favicon.ico
  - Global.asax
  - packages.config
  - Web.config



Entity Data Model Wizard

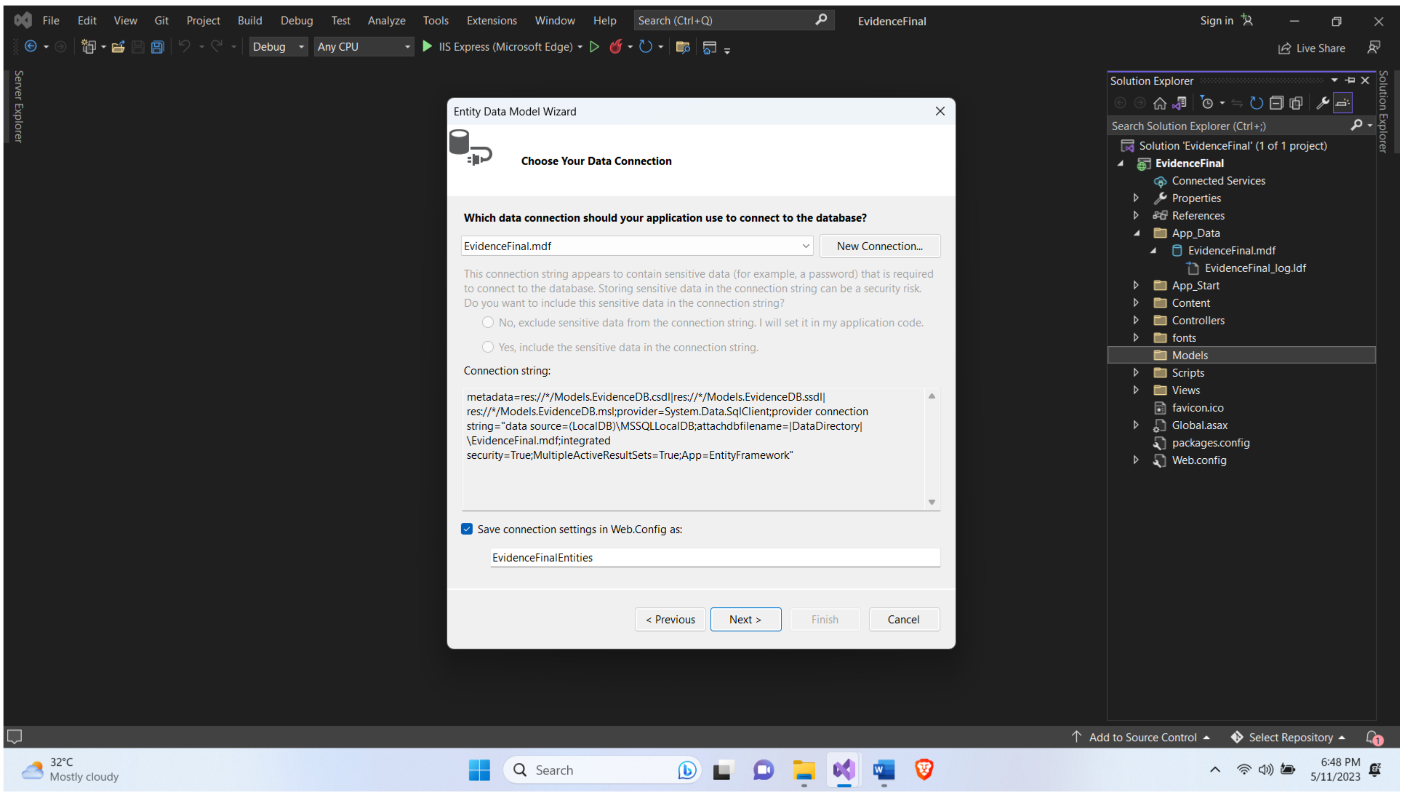
Choose Model Contents

What should the model contain?

EF Designer from database | Empty EF Designer model | Empty Code First model | Code First from database

Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.

< Previous | Next > | Finish | Cancel



Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

EvidenceFinal.mdf | New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- No, exclude sensitive data from the connection string. I will set it in my application code.
- Yes, include the sensitive data in the connection string.

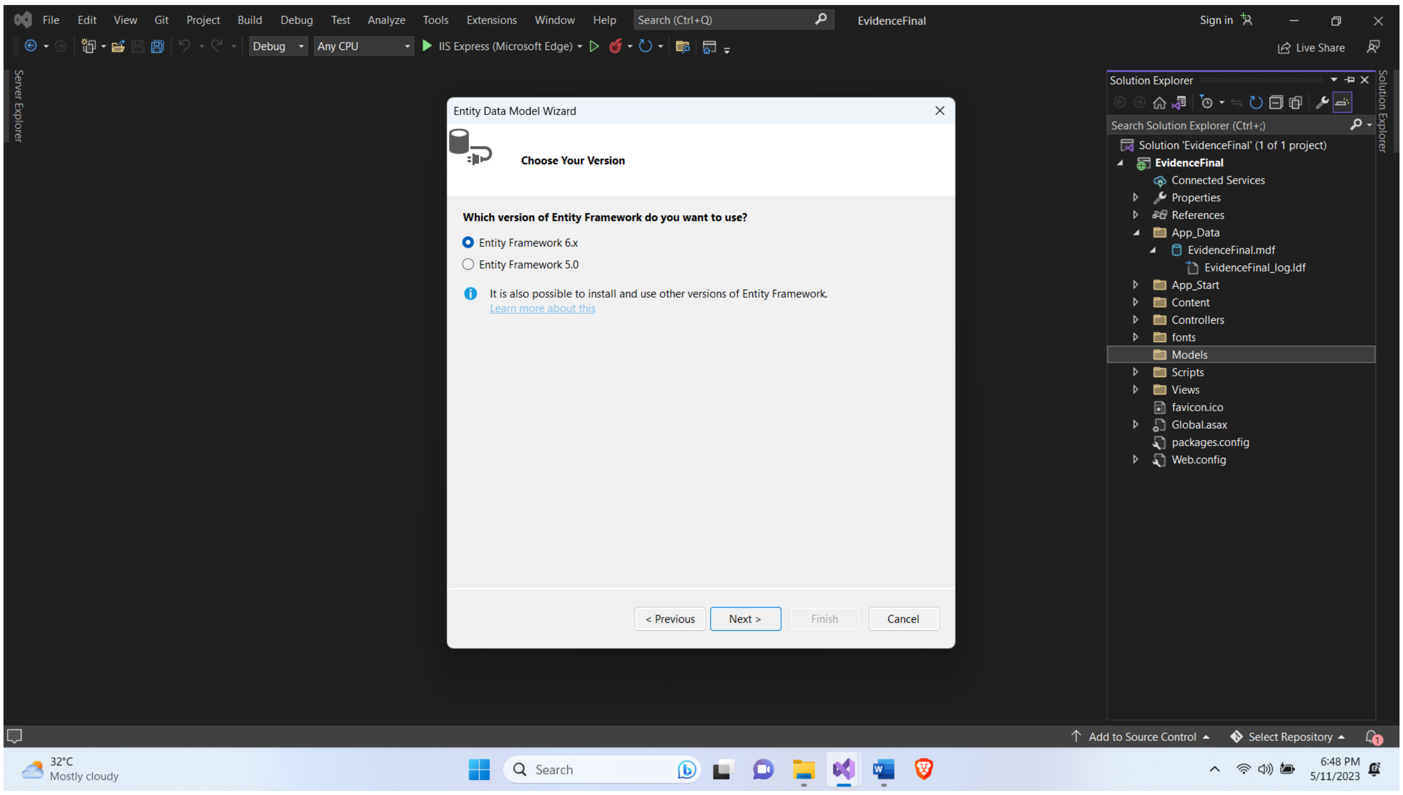
Connection string:

```
metadata=res://*/Models.EvidenceDB.csdl|res://*/Models.EvidenceDB.ssdl|
res://*/Models.EvidenceDB.msl;provider=System.Data.SqlClient;provider connection
string="data source=(LocalDB)\MSSQLLocalDB;attachdbfilename=|DataDirectory|
\EvidenceFinal.mdf;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

[✓] Save connection settings in Web.Config as:

EvidenceFinalEntities

< Previous | Next > | Finish | Cancel


Entity Data Model Wizard

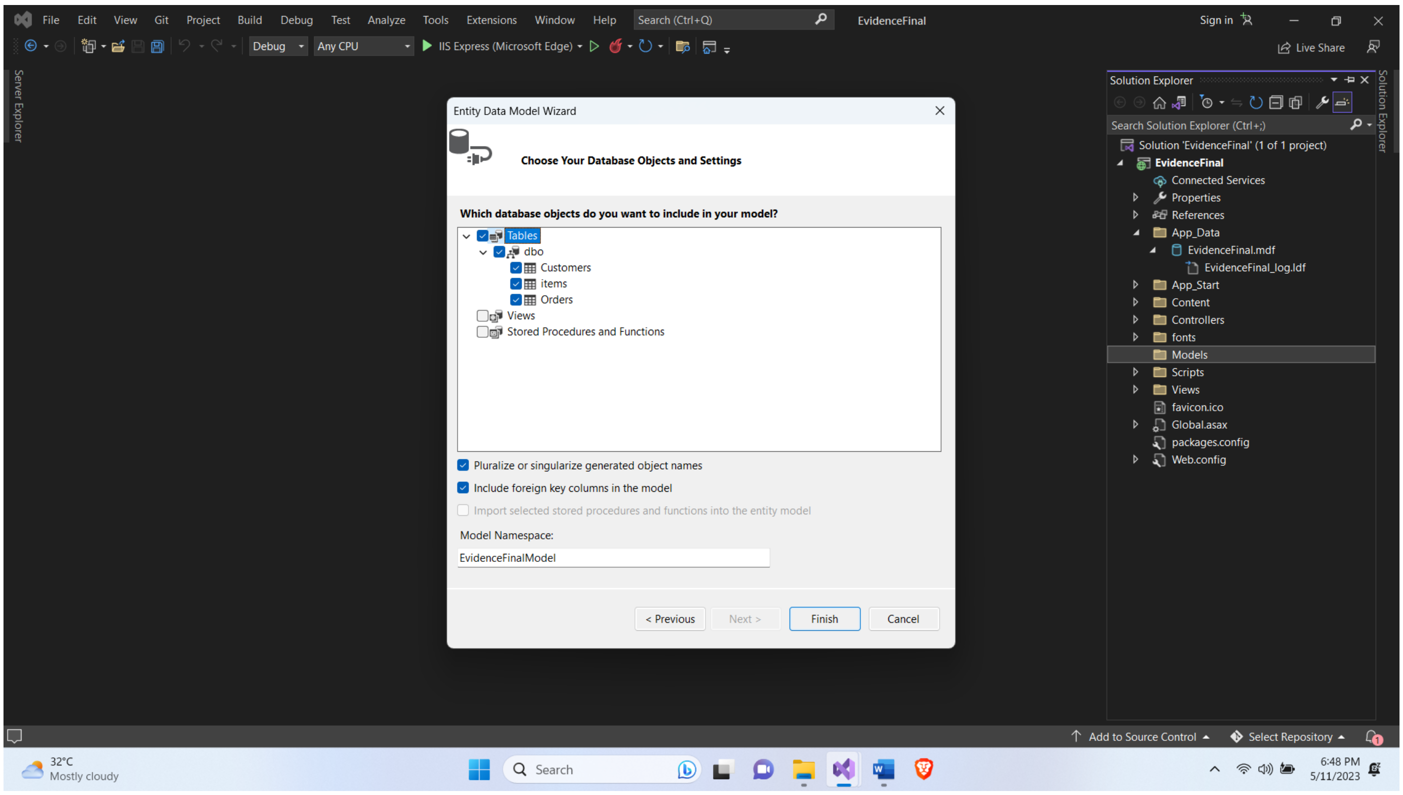
Choose Your Version

Which version of Entity Framework do you want to use?

[✓ Entity Framework 6.x]
Entity Framework 5.0

It is also possible to install and use other versions of Entity Framework.
Learn more about this

< Previous | Next > | Finish | Cancel


Entity Data Model Wizard

Choose Your Database Objects and Settings

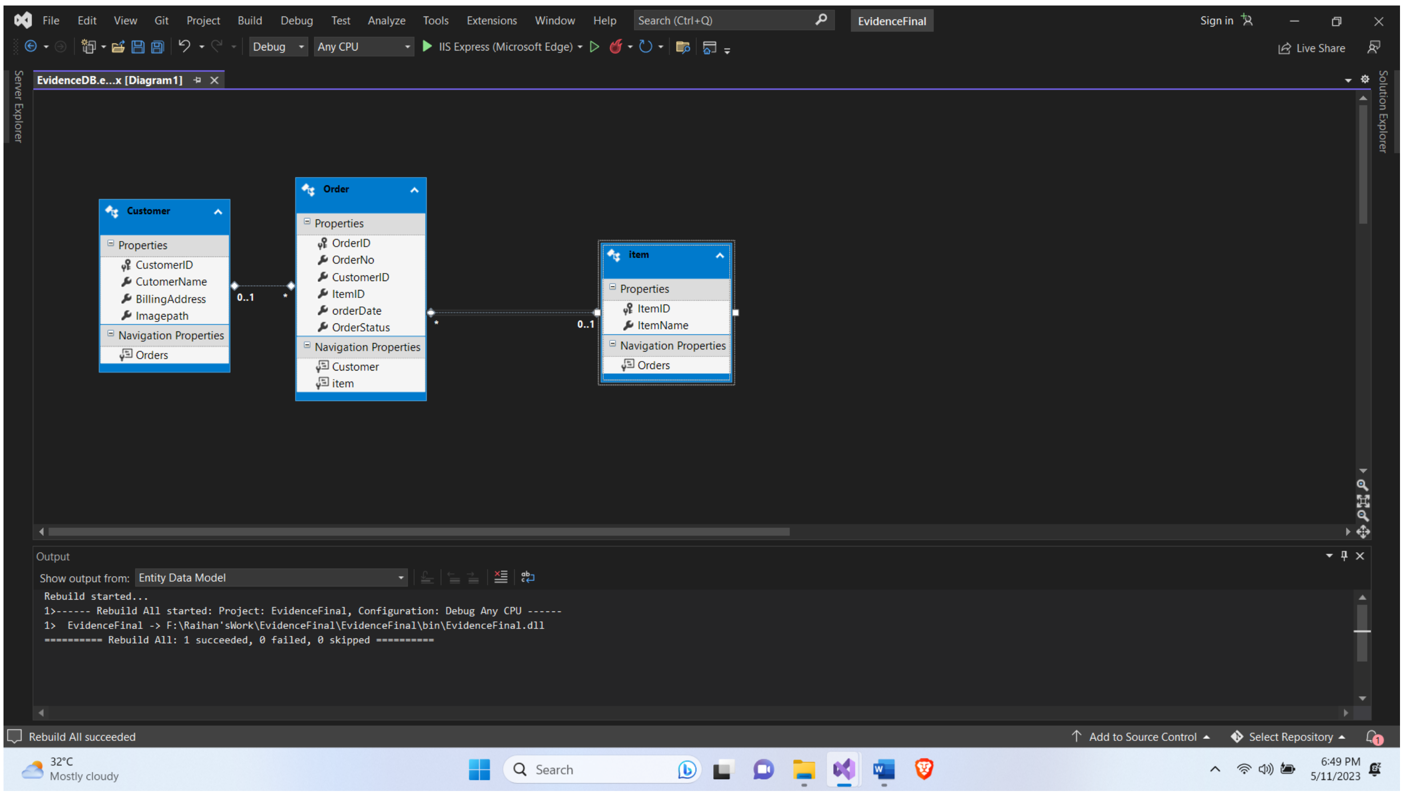
Which database objects do you want to include in your model?

- [✓] Tables
  - [✓] dbo
    - [✓] Customers
    - [✓] items
    - [✓] Orders
- [ ] Views
- [ ] Stored Procedures and Functions

[✓] Pluralize or singularize generated object names
[✓] Include foreign key columns in the model
[ ] Import selected stored procedures and functions into the entity model

Model Namespace:
EvidenceFinalModel

< Previous | Next > | Finish | Cancel

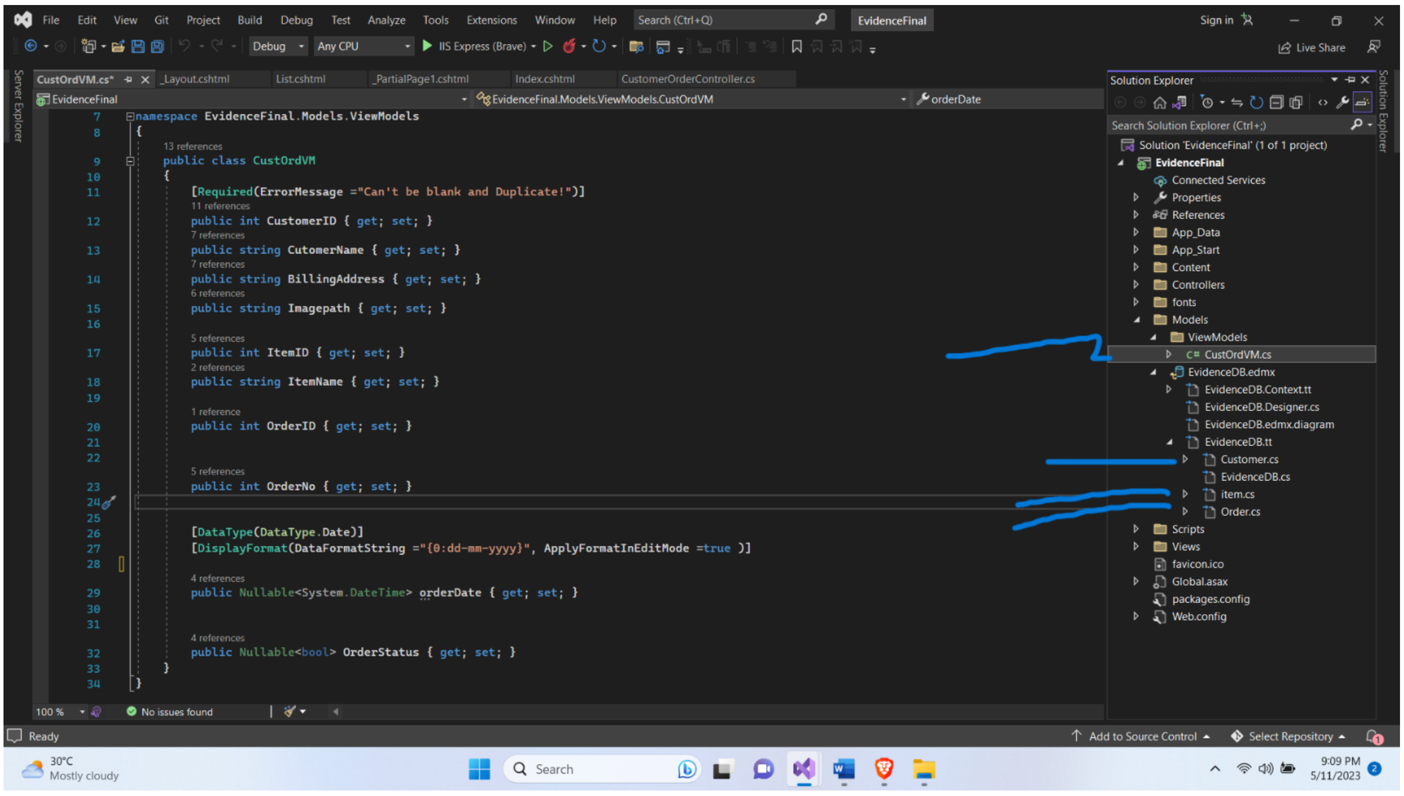


Show output from: Entity Data Model

```
Rebuild started...
1>------ Rebuild All started: Project: EvidenceFinal, Configuration: Debug Any CPU ------
1>  EvidenceFinal -> F:\Raihan'sWork\EvidenceFinal\EvidenceFinal\bin\EvidenceFinal.dll
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

Rebuild All succeeded

4. Add a **ViewModels** Folder in **Models** Folder then add a class in **ViewModels** Folder with name **CustOrdVM**
5. In **CustOrdVM** class add all the properties from **Customes.cs, items.cs, Orders.cs** . Example-

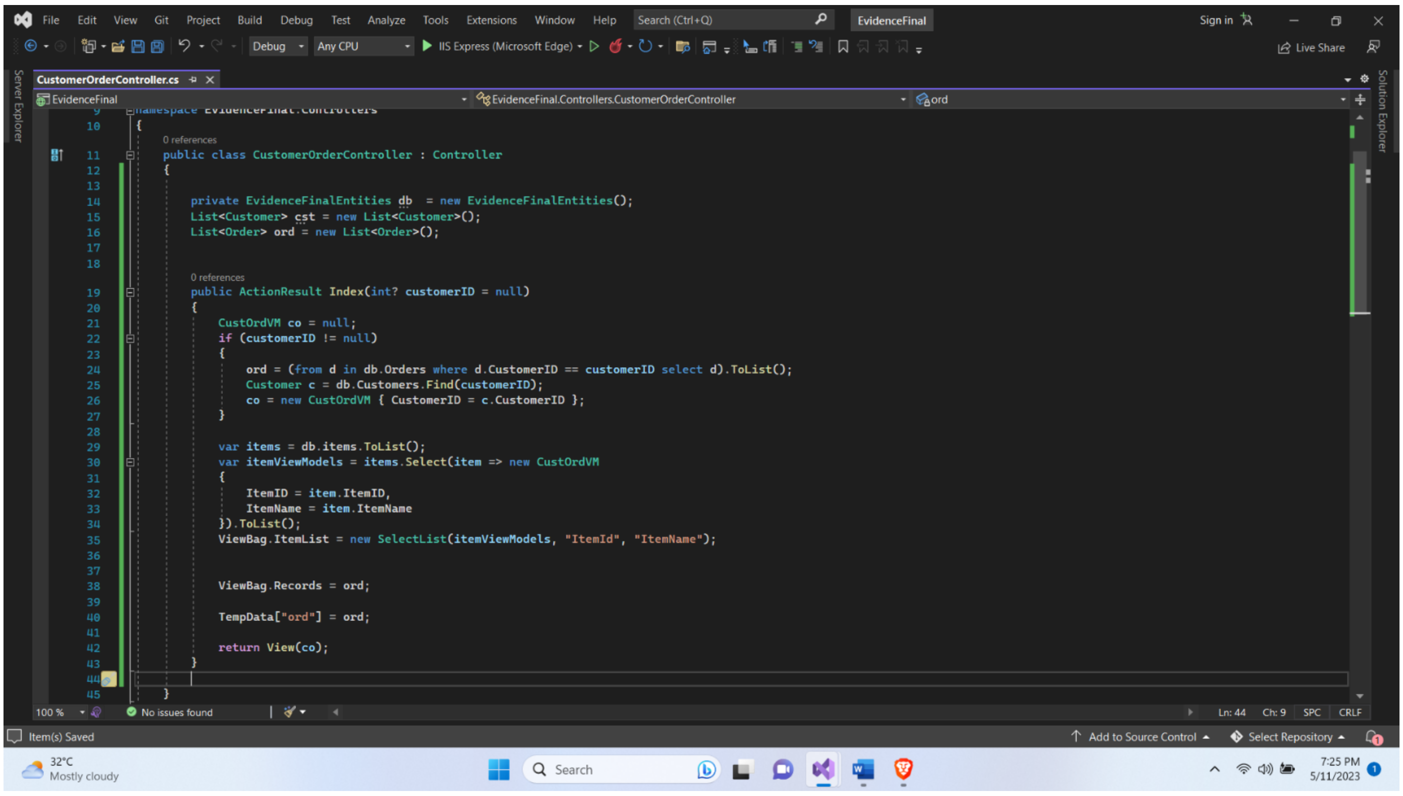


```
namespace EvidenceFinal.Models.ViewModels
{
    public class CustOrdVM
    {
        [Required(ErrorMessage ="Can't be blank and Duplicate!")]
        public int CustomerID { get; set; }
        public string CutomerName { get; set; }
        public string BillingAddress { get; set; }
        public string Imagepath { get; set; }

        public int ItemID { get; set; }
        public string ItemName { get; set; }

        public int OrderID { get; set; }


        public int OrderNo { get; set; }


        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString ="{0:dd-mm-yyyy}", ApplyFormatInEditMode =true )]

        public Nullable<System.DateTime> orderDate { get; set; }


        public Nullable<bool> OrderStatus { get; set; }
    }
}
```

No issues found

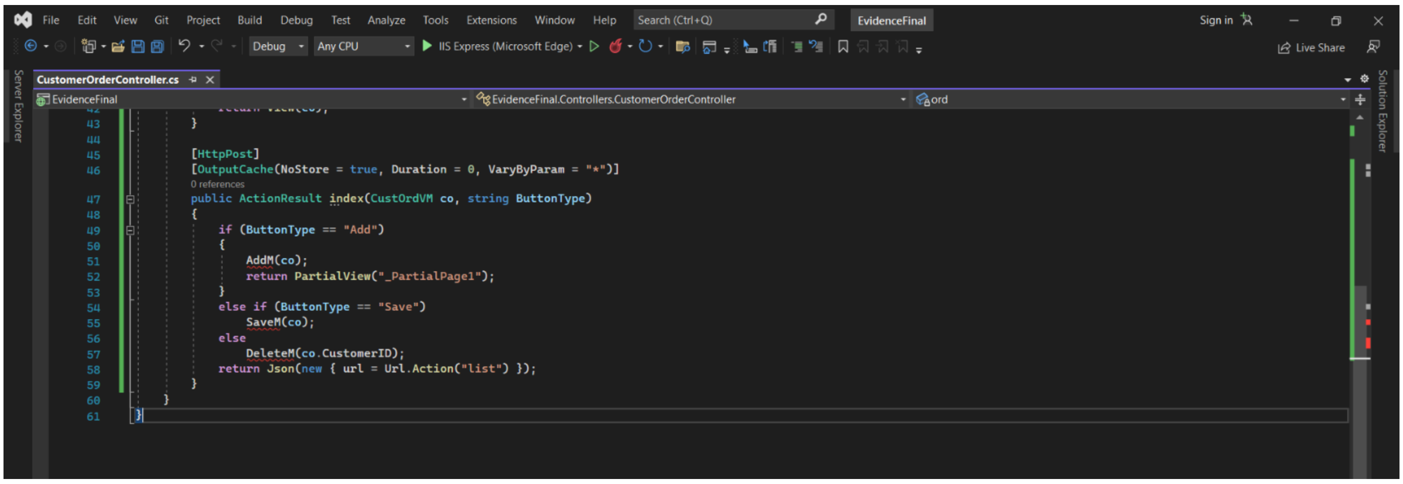
Ready

6. Add an Empty **Controller** in **Controllers** folder with the name **CustomerOrderController**
7. Add database entities and **Index** Action in this controller to **get** data from customer like this.



```
namespace EvidenceFinal.Controllers
{
    0 references
    public class CustomerOrderController : Controller
    {

        private EvidenceFinalEntities db  = new EvidenceFinalEntities();
        List<Customer> cst = new List<Customer>();
        List<Order> ord = new List<Order>();


        0 references
        public ActionResult Index(int? customerID = null)
        {
            CustOrdVM co = null;
            if (customerID != null)
            {
                ord = (from d in db.Orders where d.CustomerID == customerID select d).ToList();
                Customer c = db.Customers.Find(customerID);
                co = new CustOrdVM { CustomerID = c.CustomerID };
            }

            var items = db.items.ToList();
            var itemViewModels = items.Select(item => new CustOrdVM
            {
                ItemID = item.ItemID,
                ItemName = item.ItemName
            }).ToList();
            ViewBag.ItemList = new SelectList(itemViewModels, "ItemId", "ItemName");


            ViewBag.Records = ord;

            TempData["ord"] = ord;

            return View(co);
        }

    }
```

8. Add another Index action to post data as like this. ***(Ignore the error for AddM SaveM DeleteM in the following image. We will create this methods after few steps in this controller)***



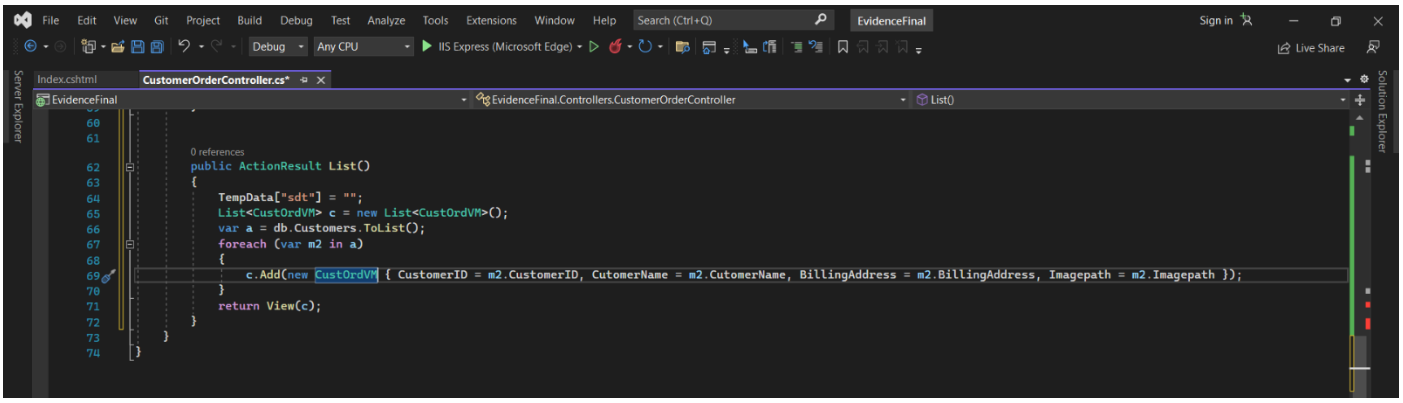
```
        }

        [HttpPost]
        [OutputCache(NoStore = true, Duration = 0, VaryByParam = "*")]
        0 references
        public ActionResult index(CustOrdVM co, string ButtonType)
        {
            if (ButtonType == "Add")
            {
                AddM(co);
                return PartialView("_PartialPage1");
            }
            else if (ButtonType == "Save")
                SaveM(co);
            else
                DeleteM(co.CustomerID);
            return Json(new { url = Url.Action("list") });
        }
    }
}
```

9. Now right click on any **index** action and **Add a View (Empty without model).**
10. Write the following code in **Index.Cshtm**l file (Note: must add viewmodel class on top **CustOrdVM**):

```
@model EvidenceFinal.Models.ViewModels.CustOrdVM
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<script>
    var onSuccess = function (result) {
        if (result.url) {
            window.location = result.url;
        }
    }
</script>


<h2>Index</h2>


@using (Ajax.BeginForm("Index", "Customer", new AjaxOptions
{
    UpdateTargetId = "table-container",
    InsertionMode = System.Web.Mvc.Ajax.InsertionMode.Replace,
    HttpMethod = "POST",
    OnSuccess = "onSuccess",
    OnFailure = "createfail"
}))

{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>CustomerOrder</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.CustomerID, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.CustomerID, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.CustomerID, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.CutomerName, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.CutomerName, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.CutomerName, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BillingAddress, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BillingAddress, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BillingAddress, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ItemName, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">

                @Html.DropDownListFor(model => model.ItemID, (SelectList)ViewBag.ItemList,
"Select Item", new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.ItemID, "", new { @class =
"text-danger" })
            </div>
        </div>



        <div class="form-group">
            @Html.LabelFor(model => model.Imagepath, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                <input type="file" name="Imagepath" id="Imagepath" class="form-control" />
                @Html.ValidationMessageFor(model => model.Imagepath, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.OrderNo, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.OrderNo, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.OrderNo, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.orderDate, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.orderDate, new { htmlAttributes = new {
@class = "form-control", type = "date" } })
                @Html.ValidationMessageFor(model => model.orderDate, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.OrderStatus, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">

                @Html.EditorFor(model => model.OrderStatus, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.OrderStatus, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Add" class="btn btn-default" name="ButtonType"
/>
            </div>
        </div>
    </div>
    <div class="col-sm-12">
        <div id="table-container">
            @Html.Partial("_PartialPage1")
        </div>
        <div>
            <input type="submit" value="Save" name="ButtonType" />
            <input type="submit" value="Delete" name="ButtonType" />
        </div>
    </div>

}

<div>
    @Html.ActionLink("Back to List", "list")
```

```
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

11. Add another List Action in same controller to view data after insert.



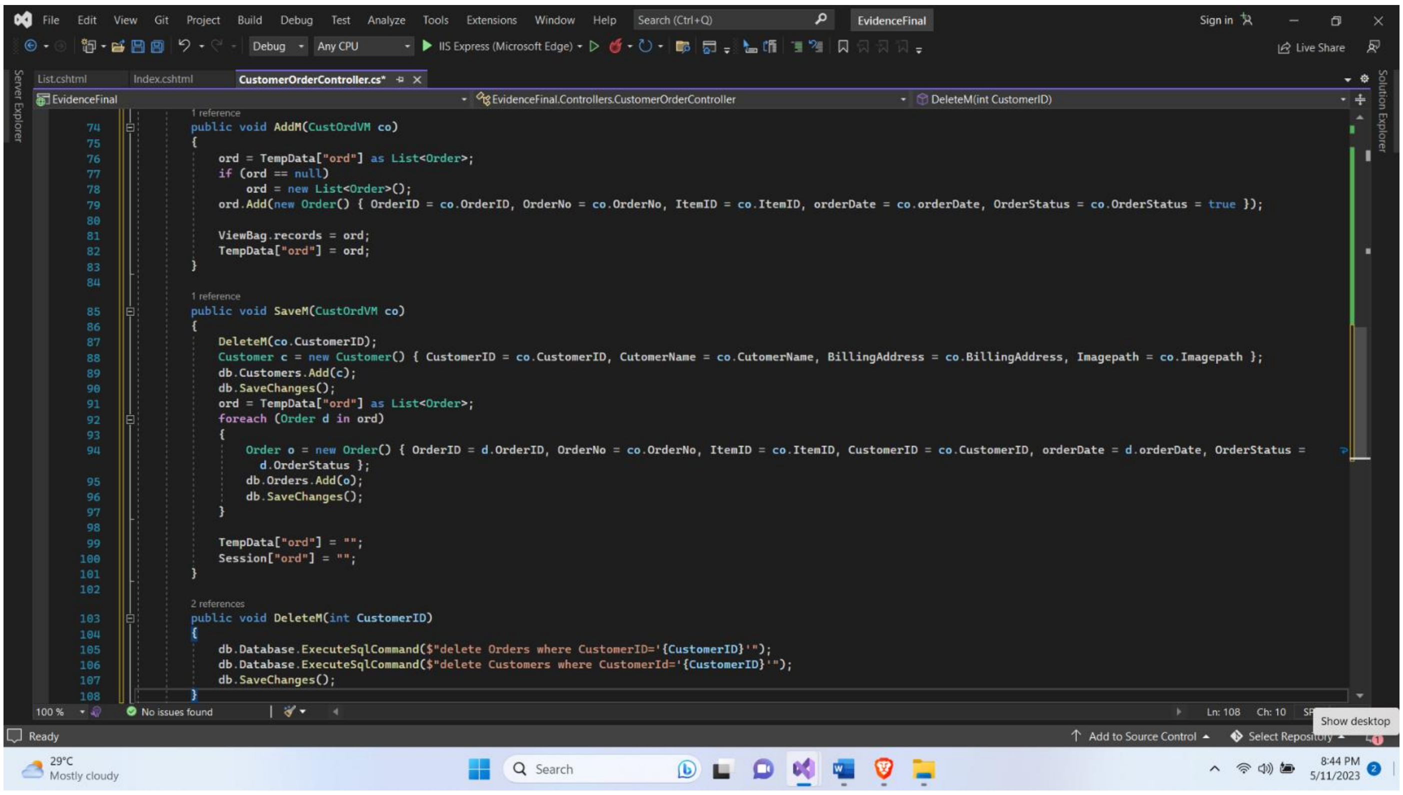
```
        public ActionResult List()
        {
            TempData["sdt"] = "";
            List<CustOrdVM> c = new List<CustOrdVM>();
            var a = db.Customers.ToList();
            foreach (var m2 in a)
            {
                c.Add(new CustOrdVM { CustomerID = m2.CustomerID, CutomerName = m2.CutomerName, BillingAddress = m2.BillingAddress, Imagepath = m2.Imagepath });
            }
            return View(c);
        }
    }
}
```

12. Right Click on List Action and **Add a View (Empty without model).**
13. Write the code in **List.Cshtml** file (Note: must add viewmodel class on top **CustOrdVM**):

```
@model IEnumerable<EvidenceFinal.Models.ViewModels.CustOrdVM>

@{
    ViewBag.Title = "List";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>List</h2>

<p>
    @Html.ActionLink("Create New", "Index")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.CutomerName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BillingAddress)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Imagepath)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.CutomerName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BillingAddress)
            </td>
            <td>
```

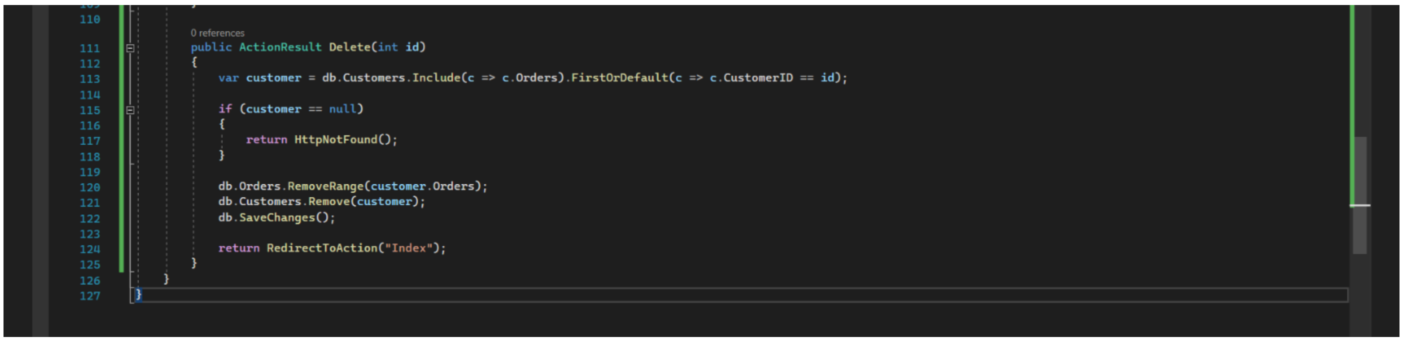
```
                @Html.DisplayFor(modelItem => item.Imagepath)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.CustomerID }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.CustomerID }, new {
onclick = "return confirm('Are you sure you want to delete this record?');" })
            </td>
        </tr>
    }

</table>
```

14. Now add those methods that are called in **index action** which is added for post.



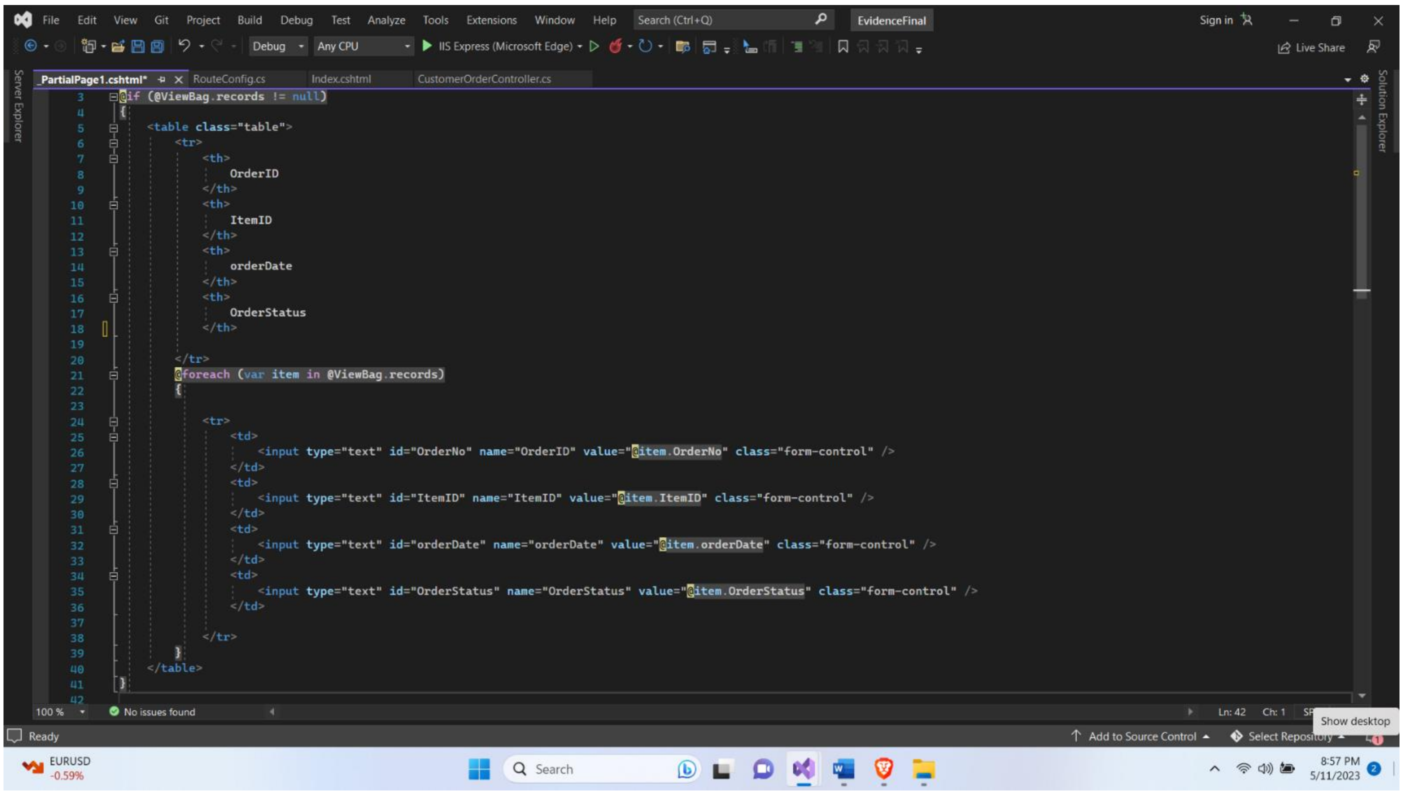
```
        public void AddM(CustOrdVM co)
        {
            ord = TempData["ord"] as List<Order>;
            if (ord == null)
                ord = new List<Order>();
            ord.Add(new Order() { OrderID = co.OrderID, OrderNo = co.OrderNo, ItemID = co.ItemID, orderDate = co.orderDate, OrderStatus = co.OrderStatus = true });

            ViewBag.records = ord;
            TempData["ord"] = ord;
        }

        public void SaveM(CustOrdVM co)
        {
            DeleteM(co.CustomerID);
            Customer c = new Customer() { CustomerID = co.CustomerID, CutomerName = co.CutomerName, BillingAddress = co.BillingAddress, Imagepath = co.Imagepath };
            db.Customers.Add(c);
            db.SaveChanges();
            ord = TempData["ord"] as List<Order>;
            foreach (Order d in ord)
            {
                Order o = new Order() { OrderID = d.OrderID, OrderNo = co.OrderNo, ItemID = co.ItemID, CustomerID = co.CustomerID, orderDate = d.orderDate, OrderStatus =
                  d.OrderStatus };
                db.Orders.Add(o);
                db.SaveChanges();
            }

            TempData["ord"] = "";
            Session["ord"] = "";
        }

        public void DeleteM(int CustomerID)
        {
            db.Database.ExecuteSqlCommand($"delete Orders where CustomerID='{CustomerID}'");
            db.Database.ExecuteSqlCommand($"delete Customers where CustomerId='{CustomerID}'");
            db.SaveChanges();
        }
```

15. Add another **Delete action** in this controller for cascade delete. Note: We called this action in List.Cshtml file



```
        public ActionResult Delete(int id)
        {
            var customer = db.Customers.Include(c => c.Orders).FirstOrDefault(c => c.CustomerID == id);

            if (customer == null)
            {
                return HttpNotFound();
            }

            db.Orders.RemoveRange(customer.Orders);
            db.Customers.Remove(customer);
            db.SaveChanges();

            return RedirectToAction("Index");
        }
    }
}
```

16. Now add a **PartialView** page in **Views > Shared folder** and write this code in _PartialView.Cshtml file. Note that we already declare this page in Index action (post) and also in Index.Cshtml page –

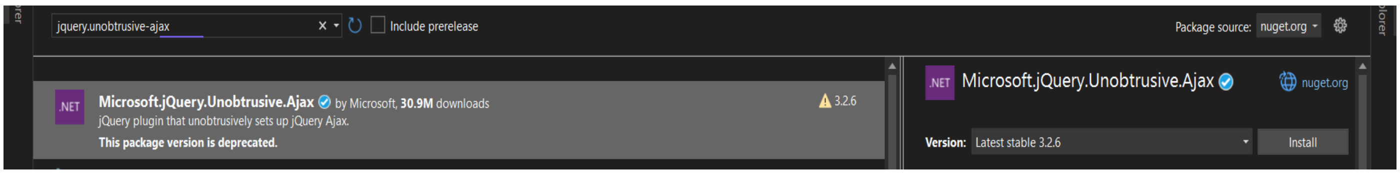


```
@if (@ViewBag.records != null)
{
    <table class="table">
        <tr>
            <th>
                OrderID
            </th>
            <th>
                ItemID
            </th>
            <th>
                orderDate
            </th>
            <th>
                OrderStatus
            </th>

        </tr>
        @foreach (var item in @ViewBag.records)
        {

            <tr>
                <td>
                    <input type="text" id="OrderNo" name="OrderID" value="@item.OrderNo" class="form-control" />
                </td>
                <td>
                    <input type="text" id="ItemID" name="ItemID" value="@item.ItemID" class="form-control" />
                </td>
                <td>
                    <input type="text" id="orderDate" name="orderDate" value="@item.orderDate" class="form-control" />
                </td>
                <td>
                    <input type="text" id="OrderStatus" name="OrderStatus" value="@item.OrderStatus" class="form-control" />
                </td>

            </tr>
        }
    </table>
}
```

17. Go to RoutConfig.cs in **App Start** Folder and change the controller name to CustomerOrder and action name to Index.

18. Add Some data in items table manually..

19. Install this jquery and add it `jquery.unobtrusive-ajax.min.js` to _Layout,Cshtml File



jquery.unobtrusive-ajax
Include prerelease
Package source: nuget.org
Microsoft.jQuery.Unobtrusive.Ajax by Microsoft, 30.9M downloads
3.2.6
jQuery plugin that unobtrusively sets up jQuery Ajax.
This package version is deprecated.
Microsoft.jQuery.Unobtrusive.Ajax
nuget.org
Version: Latest stable 3.2.6
Install

20. Rebuild Your Project And Run!!