LINQ and Entity Framework

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes

Table of Contents

Getting Started

Overview

This Lab book is a guided tour for LINQ and Entity Framework 6. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for LINQ and Entity Framework 6

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

Processor, HDD & RAM

- Processor - Minimum: Intel Core 3 or Higher
- Processor speed: Minimum: 1.4 GHz
- Recommended: 2.0 GHz or faster
- RAM - Minimum:2 GB, Recommended: 4 GB or more
- HDD – 150 GB

Operating System

- Windows 7 Professional 64 bit
- Windows 10 Professional 64 bit

Please ensure that the following is done:

- Visual Studio 2017 or above

Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory html_assgn. For each lab exercise create a directory as lab <lab number>.
- Download all files required to complete assignments from: http://pace.patni.com/TechRS/download.asp?course=Internet_HTML
- You may also look up the on-line help provided in the MSDN library.

Learning More (Bibliography if applicable)

- http://msdn.microsoft.com
- http://www.asp.net/entity-framework
- https://msdn.microsoft.com/en-in/data/ef.aspx
- Entity Framework 6 Recepies by Apress publication

Lab 1. LINQ Basics

Goals	Understand the process of Implementing LINQ to a Collection Learn to use LINQ Learn to use LINQ Operators
Time	60 minutes

1. Create a console application and add class named Employee with following field.

   Employee Class

   EmployeeID (Integer)

   FirstName (String)

   LastName (String)

   Title (String)

   DOB (Date)

   DOJ (Date)

   City (String)

2. Create a Generic List Collection empList and populate it with the following records.

EmployeeID	FirstName	LastName	Title	DOB	DOJ	City
1001	Malcolm	Daruwalla	Manager	16/11/1984	8/6/2011	Mumbai
1002	Asdin	Dhalla	AsstManager	20/08/1984	7/7/2012	Mumbai
1003	Madhavi	Oza	Consultant	14/11/1987	12/4/2015	Pune
1004	Saba	Shaikh	SE	3/6/1990	2/2/2016	Pune
1005	Nazia	Shaikh	SE	8/3/1991	2/2/2016	Mumbai
1006	Amit	Pathak	Consultant	7/11/1989	8/8/2014	Chennai
1007	Vijay	Natrajan	Consultant	2/12/1989	1/6/2015	Mumbai
1008	Rahul	Dubey	Associate	11/11/1993	6/11/2014	Chennai
1009	Suresh	Mistry	Associate	12/8/1992	3/12/2014	Chennai
1010	Sumit	Shah	Manager	12/4/1991	2/1/2016	Pune

3. Now once the collection created write down and execute the LINQ queries for collection as follows
   a. Display detail of all the employee
   b. Display details of all the employee whose location is not Mumbai
   c. Display details of all the employee whose title is AsstManager
   d. Display details of all the employee whose Last Name start with S
   e. Display a list of all the employee who have joined before 1/1/2015

f. Display a list of all the employee whose date of birth is after 1/1/1990
g. Display a list of all the employee whose designation is Consultant and Associate
h. Display total number of employees
i. Display total number of employees belonging to “Chennai”
j. Display highest employee id from the list
k. Display total number of employee who have joined after 1/1/2015
l. Display total number of employee whose designation is not “Associate”
m. Display total number of employee based on City
n. Display total number of employee based on city and title
o. Display total number of employee who is youngest in the list

Lab 2. Using LINQ to DataSet / DataTable

Goals	Implementing LINQ to Dataset or LINQ to Datatable
Time	45 minutes

Create table & insert records based on following SQL Script provided.

```
create table Employee(
        EmployeeID int,
        FirstName varchar(20),
        LastName varchar(20),
        Title varchar(20),
        DOB datetime,
        DOJ datetime,
        City varchar(20)
);

insert into Employee values
(1001, 'Malcolm', 'Daruwalla','Manager','11/16/1984','6/8/2011','Mumbai'),
(1002,'Asdin','Dhalla','AsstManager','08/20/1984','7/7/2012','Mumbai'),
(1003,'Madhavi','Oza','Consultant','11/14/1987','4/12/2015','Pune'),
(1004,'Saba', 'Shaikh','SE','6/3/1990','2/2/2016','Pune'),
(1005,'Nazia', 'Shaikh','SE','3/8/1991','2/2/2016','Mumbai'),
(1006,'Suresh', 'Pathak','Consultant','11/7/1989','8/8/2014','Chennai'),
(1007,'Vijay', 'Natrajan','Consultant','12/2/1989','6/1/2015','Mumbai'),
(1008,'Rahul', 'Dubey','Associate','11/11/1993','11/6/2014','Chennai'),
(1009,'Amit', 'Mistry','Associate','8/12/1992','12/3/2014','Chennai'),
(1010,'Sumit', 'Shah','Manager','4/12/1991','1/2/2016','Pune');

select * from Employee
```

1. Write LINQ to DataSet Queries for following requirements
    a. Sort & Display Employee by their Joining Year in ascending order & then by their First-Name in descending order.
    b. Get Average Age in Years (Eg-32.87) of Manager Employees
    c. Group Employees by their City in such way that younger Employees will be displayed on top

Lab 3. Creating Entity Data Model

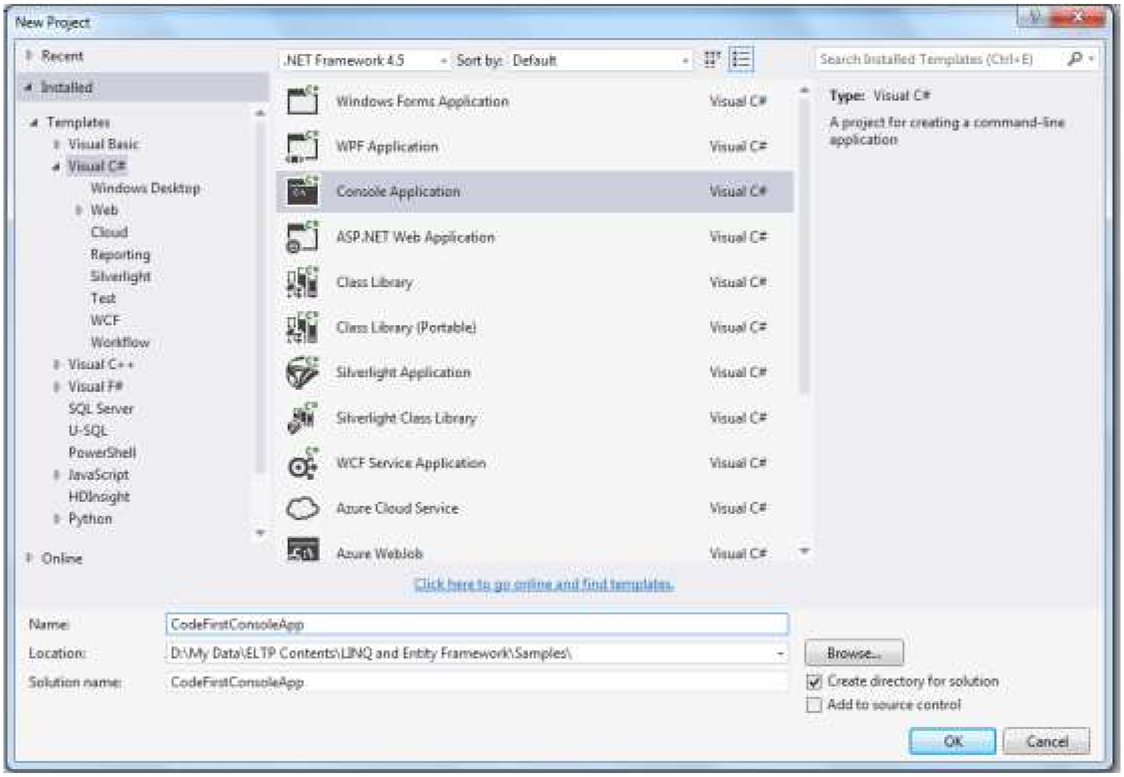
Goals	Understand the process of Creating Entity Data Model Learn to use Code First Approach Learn to use Database First Approach
Time	60 minutes

Part 1:-

Using Code-First approach

Solution:-

Create a Console application and name the application as CodeFirstConsoleApp


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows Desktop
Web
Cloud
Reporting
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
U-SQL
PowerShell
JavaScript
HDInsight
Python
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
WCF Service Application Visual C#
Azure Cloud Service Visual C#
Azure WebJob Visual C#
Type: Visual C#
A project for creating a command-line application
Click here to go online and find templates.
Browse...
OK
Cancel

Name: CodeFirstConsoleApp

Location: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\

Solution name: CodeFirstConsoleApp

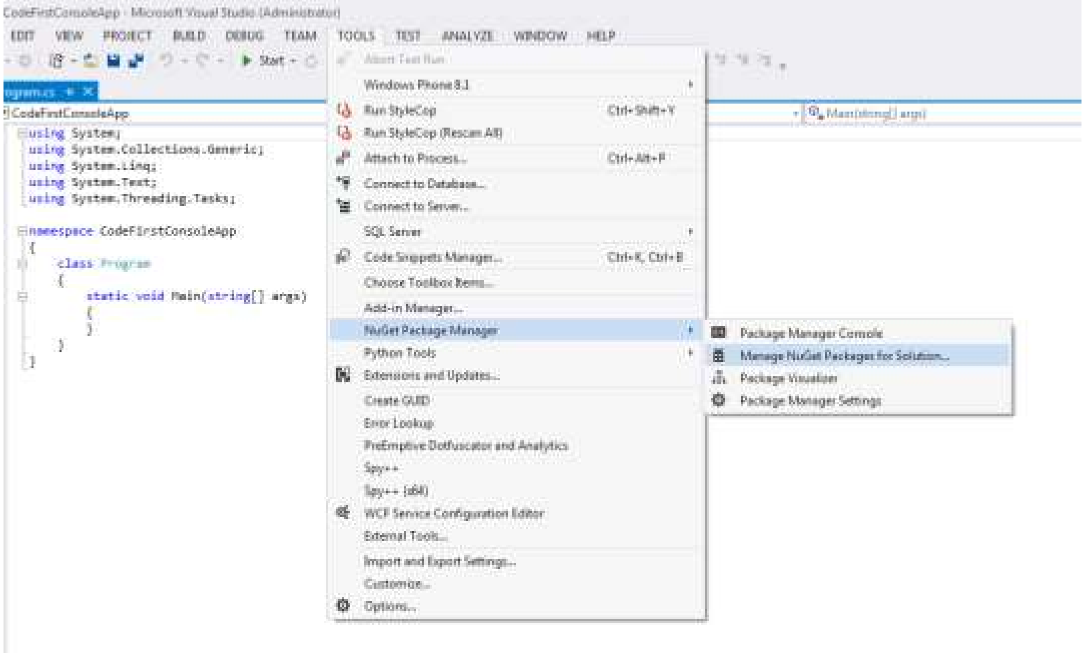
☑ Create directory for solution

□ Add to source control

After the project is created Now we have to add the Entity framework Library to Project . For that we will Nuget Package Manager Dialog

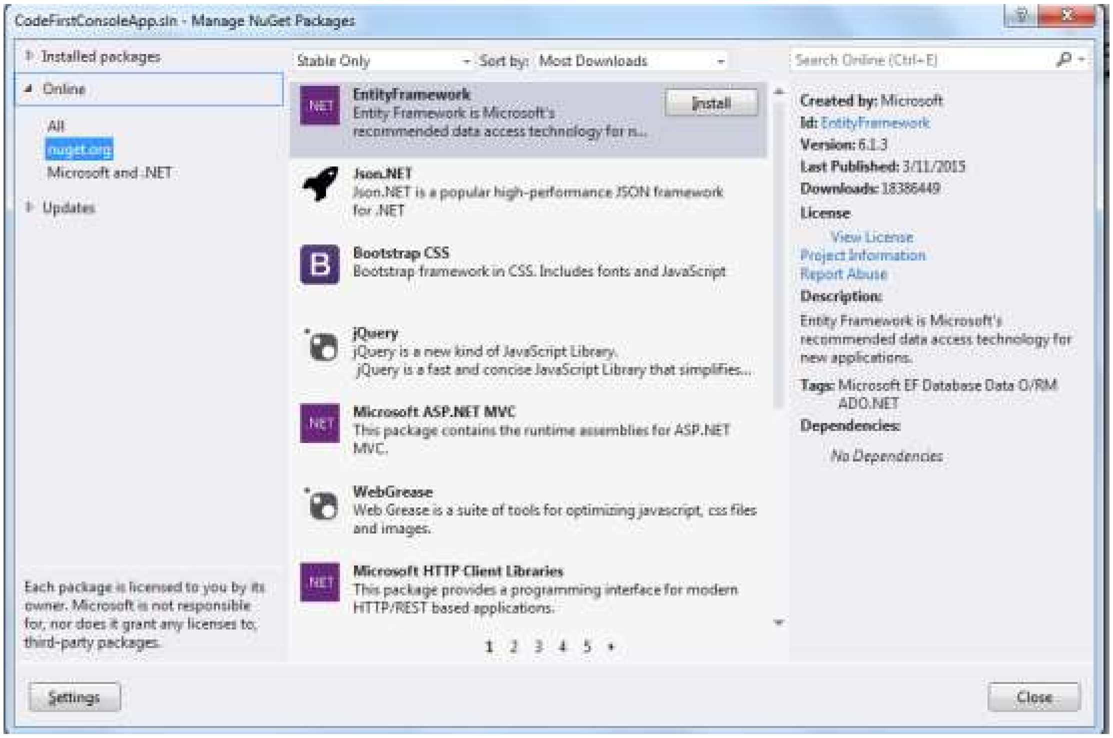
To Open Nuget Package Manager Dialog we need to follow the following step

Tools -> Nuget Package Manager -> Manage Nuget Package for the Solution


CodeFirstConsoleApp - Microsoft Visual Studio (Administrator)
EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Start
ogram.cs
CodeFirstConsoleApp
Main(string[] args)
Attach Test Run
Windows Phone 8.1
Run StyleCop Ctrl+Shift+Y
Run StyleCop (Rescan All)
Attach to Process... Ctrl+Alt+P
Connect to Database...
Connect to Server...
SQL Server
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
Add-in Manager...
NuGet Package Manager
Python Tools
Extensions and Updates...
Create GUID
Error Lookup
PreEmptive Dotfuscator and Analytics
Spy++
Spy++ (x64)
WCF Service Configuration Editor
External Tools...
Import and Export Settings...
Customize...
Options...
Package Manager Console
Manage NuGet Packages for Solution...
Package Visualizer
Package Manager Settings

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CodeFirstConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

In the dialog box for Nuget Package Manager select entity framework and click on install this will install EntityFramework to the project



EntityFramework
Entity Framework is Microsoft's recommended data access technology for n...

Json.NET
Json.NET is a popular high-performance JSON framework for .NET

Bootstrap CSS
Bootstrap framework in CSS. Includes fonts and JavaScript

jQuery
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies...

Microsoft ASP.NET MVC
This package contains the runtime assemblies for ASP.NET MVC.

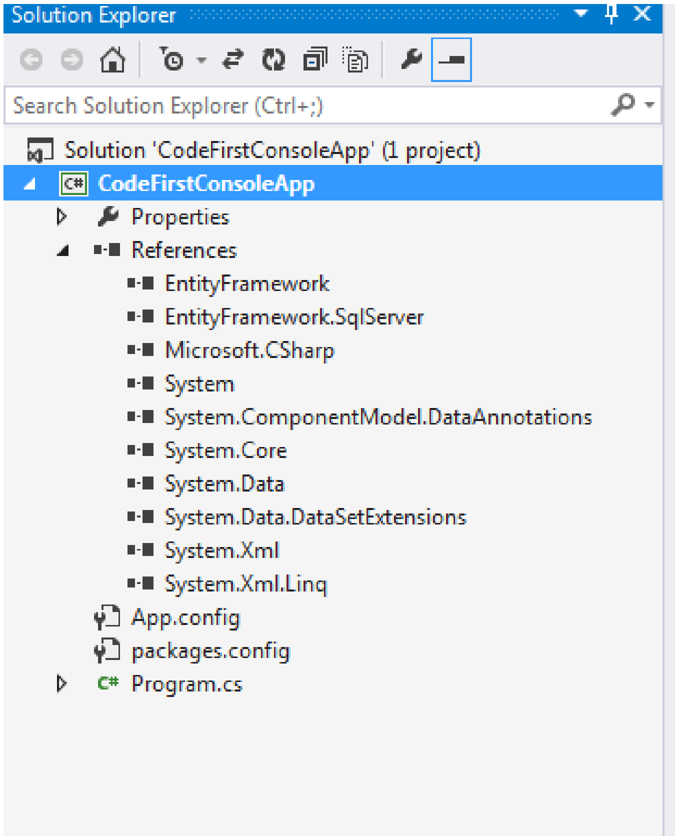
WebGrease
Web Grease is a suite of tools for optimizing javascript, css files and images.

Microsoft HTTP Client Libraries
This package provides a programming interface for modern HTTP/REST based applications.

Created by: Microsoft
Id: EntityFramework
Version: 6.1.3
Last Published: 3/11/2015
Downloads: 18386449
License
View License
Project Information
Report Abuse
Description:
Entity Framework is Microsoft's recommended data access technology for new applications.
Tags: Microsoft EF Database Data O/RM ADO.NET
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

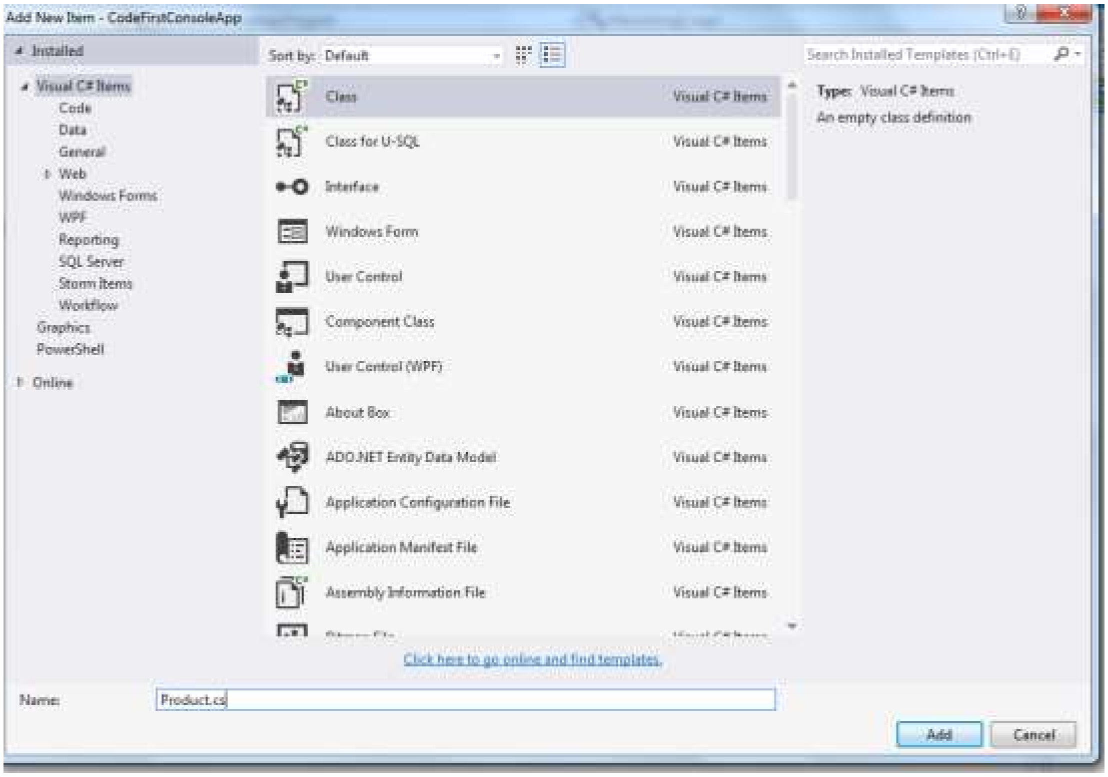
After installing the EntityFramework to the project we can see that EntityFrame dll file has been added to the References folder in Solution Explorer



```
Solution 'CodeFirstConsoleApp' (1 project)
  CodeFirstConsoleApp
    Properties
    References
      EntityFramework
      EntityFramework.SqlServer
      Microsoft.CSharp
      System
      System.ComponentModel.DataAnnotations
      System.Core
      System.Data
      System.Data.DataSetExtensions
      System.Xml
      System.Xml.Linq
    App.config
    packages.config
    Program.cs
```

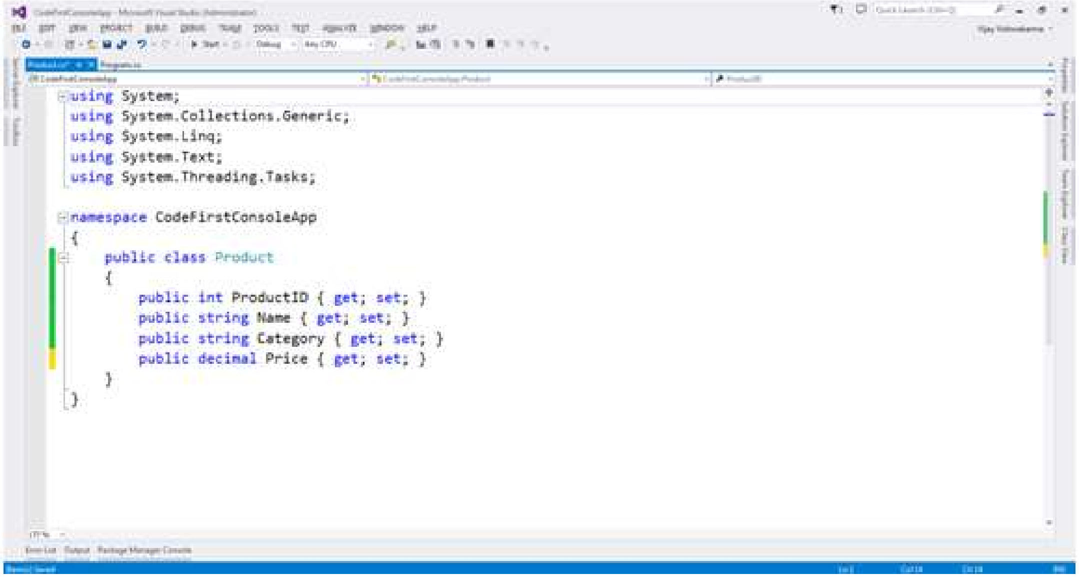
Now as we have added the EntityFramework dll to the project now we have to add the Entity and Context to the project.

To add a entity class In the Solution Explorer right click on the project name than Add □ Class and name the class as Product.cs


Add New Item - CodeFirstConsoleApp
Installed
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
SQL Server
Storm Items
Workflow
Graphics
PowerShell
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
Class — Visual C# Items
Class for U-SQL — Visual C# Items
Interface — Visual C# Items
Windows Form — Visual C# Items
User Control — Visual C# Items
Component Class — Visual C# Items
User Control (WPF) — Visual C# Items
About Box — Visual C# Items
ADO.NET Entity Data Model — Visual C# Items
Application Configuration File — Visual C# Items
Application Manifest File — Visual C# Items
Assembly Information File — Visual C# Items
Type: Visual C# Items
An empty class definition
Click here to go online and find templates.
Add
Cancel

Name: Product.cs

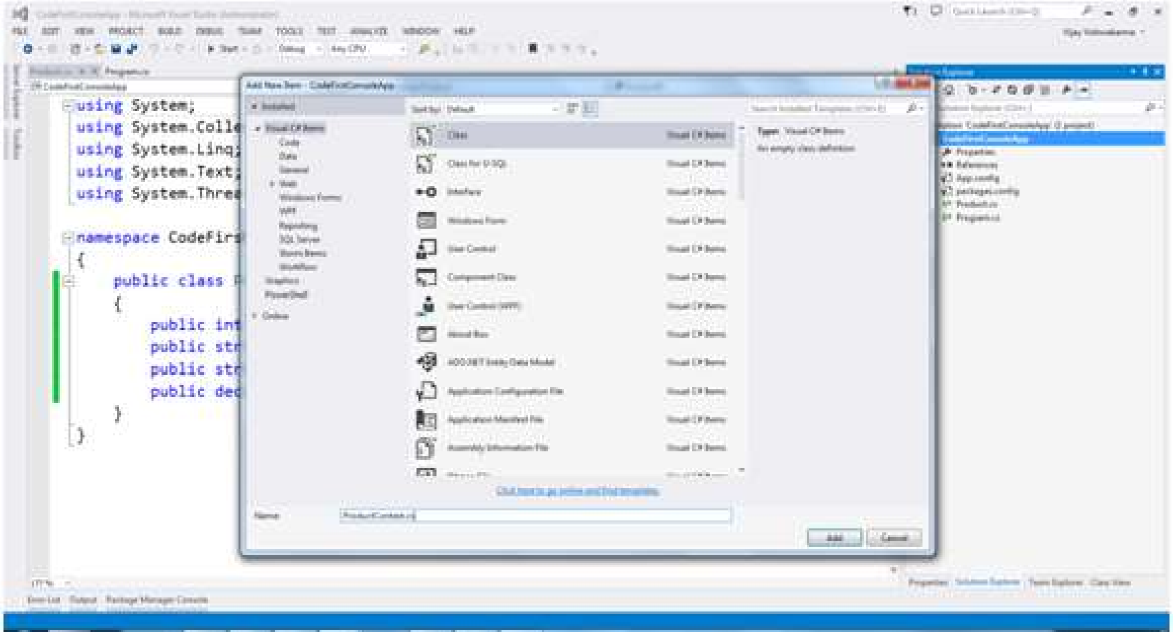
Once the class has been created add the following code to the class



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CodeFirstConsoleApp
{
    public class Product
    {
        public int ProductID { get; set; }
        public string Name { get; set; }
        public string Category { get; set; }
        public decimal Price { get; set; }
    }
}
```

After adding the class now we have to add a Context Class to the project .Context class will allow to perform database operation like add ,delete etc.

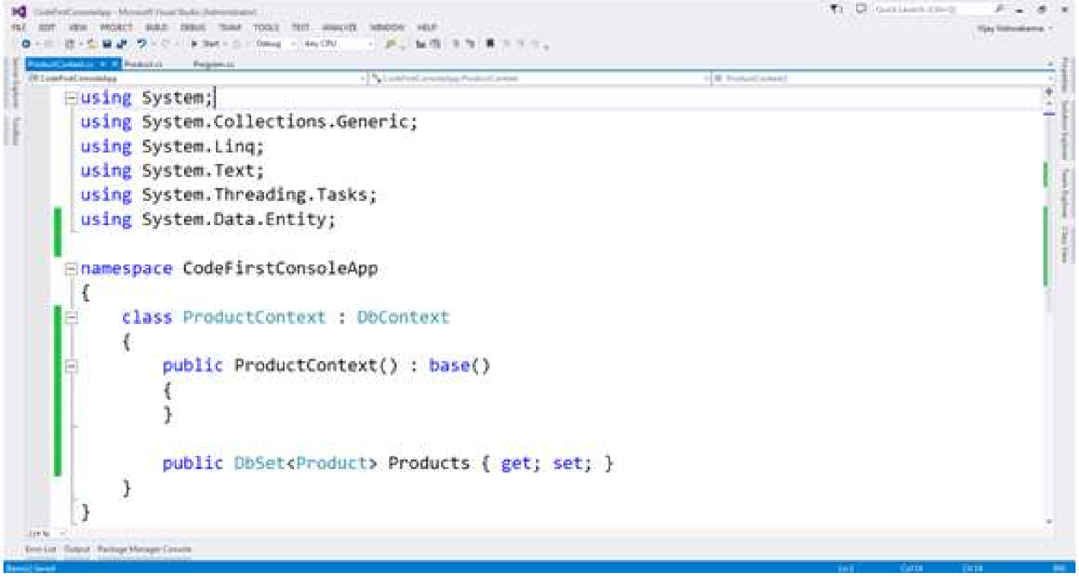
Context Class will always inherit from DbContext class available in System.Data.Entity namespace



Name: ProductContext.cs

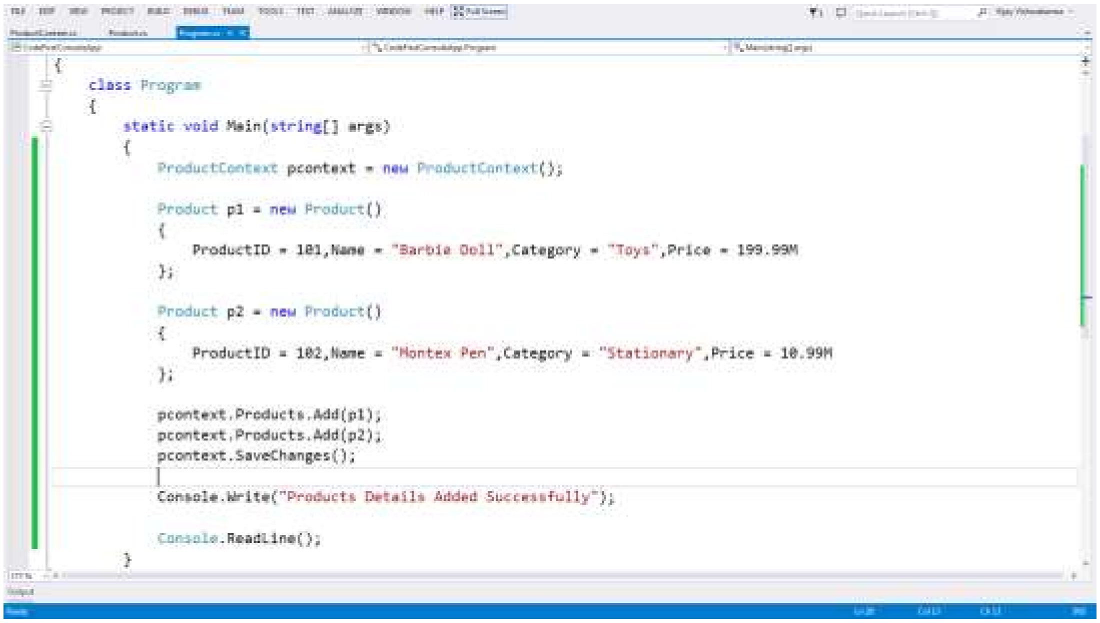
```
using System;
using System.Colle
using System.Linq;
using System.Text;
using System.Threa

namespace CodeFirs
{
    public class P
    {
        public int
        public str
        public str
        public dec
    }
}
```

After adding the context class add the following code to the class



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;

namespace CodeFirstConsoleApp
{
    class ProductContext : DbContext
    {
        public ProductContext() : base()
        {
        }

        public DbSet<Product> Products { get; set; }
    }
}
```

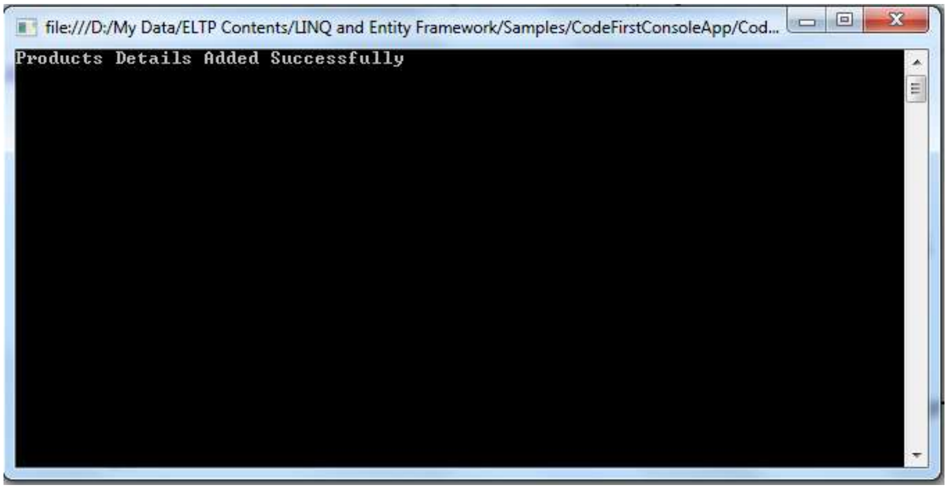
In the Program.cs class file add the following code



```
    {
        class Program
        {
            static void Main(string[] args)
            {
                ProductContext pcontext = new ProductContext();

                Product p1 = new Product()
                {
                    ProductID = 101,Name = "Barbie Doll",Category = "Toys",Price = 199.99M
                };

                Product p2 = new Product()
                {
                    ProductID = 102,Name = "Montex Pen",Category = "Stationary",Price = 10.99M
                };

                pcontext.Products.Add(p1);
                pcontext.Products.Add(p2);
                pcontext.SaveChanges();

                Console.Write("Products Details Added Successfully");

                Console.ReadLine();
            }
```

When the above code is executed it will create the database and table based on the Entity and the above record into the table

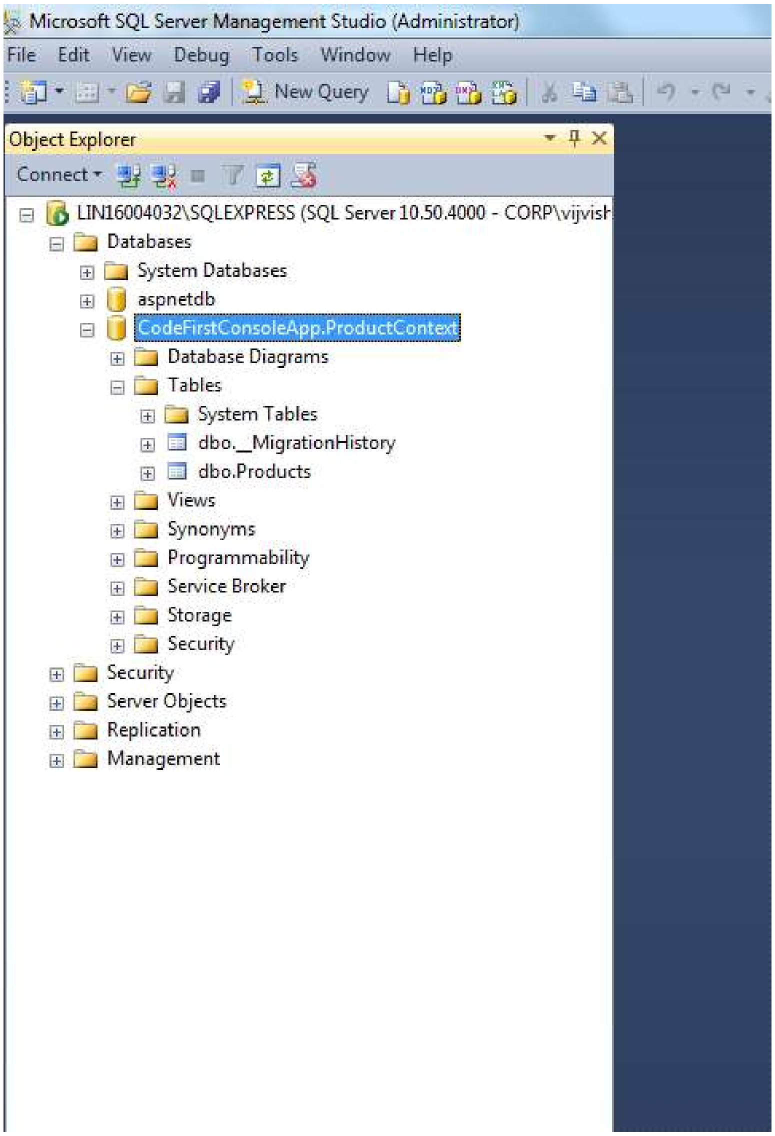
After executing the application we will get the following output



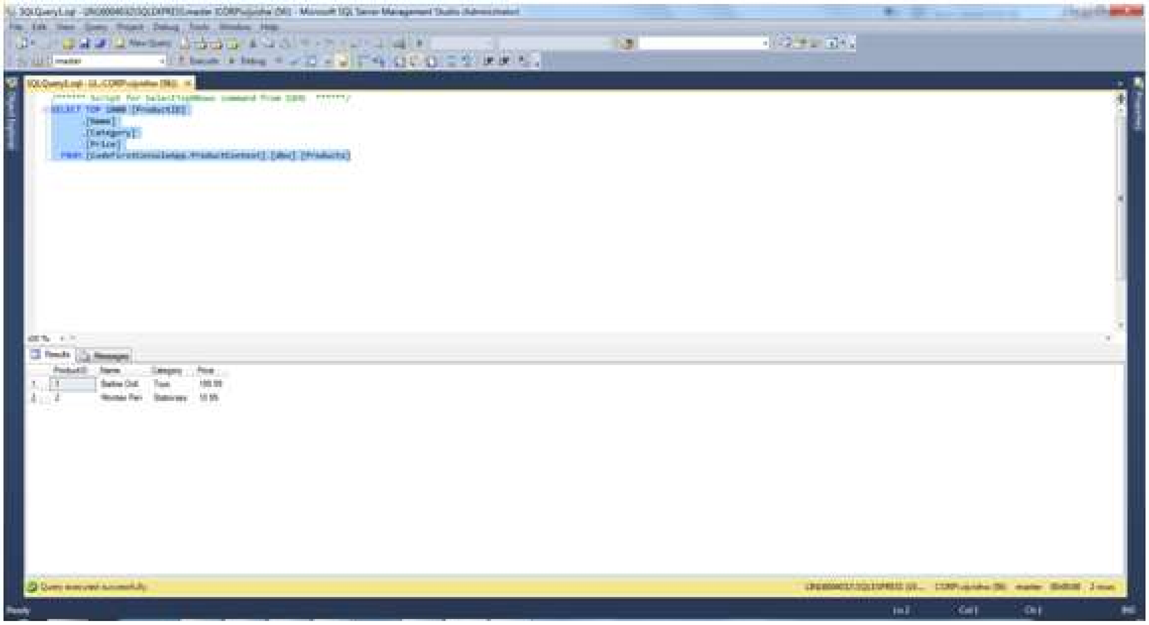
file:///D:/My Data/ELTP Contents/LINQ and Entity Framework/Samples/CodeFirstConsoleApp/Cod...

```
Products Details Added Successfully
```

Now we will check database which is created for that open Sql Server Management Studio and connect to the default instance. In the object explorer you can see the database and table being created


Microsoft SQL Server Management Studio (Administrator)
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect

```
⊟ LIN16004032\SQLEXPRESS (SQL Server 10.50.4000 - CORP\vijvish
   ⊟ Databases
      ⊞ System Databases
      ⊞ aspnetdb
      ⊟ CodeFirstConsoleApp.ProductContext
         ⊞ Database Diagrams
         ⊟ Tables
            ⊞ System Tables
            ⊞ dbo.__MigrationHistory
            ⊞ dbo.Products
         ⊞ Views
         ⊞ Synonyms
         ⊞ Programmability
         ⊞ Service Broker
         ⊞ Storage
         ⊞ Security
   ⊞ Security
   ⊞ Server Objects
   ⊞ Replication
   ⊞ Management
```



```
/****** Script for SelectTopNRows command from SSMS ******/
SELECT TOP 1000 [ProductID]
      ,[Name]
      ,[Category]
      ,[Price]
  FROM [CodeFirstConsoleApp.ProductContext].[dbo].[Products]
```

	ProductID	Name	Category	Price
1	1	Barbie Doll	Toys	199.99
2	2	Wooden Pen	Stationary	10.99

Query executed successfully.

Part 2:-

Using Database First Approach

Solution:-

Open Sql Server Management Studio and Create a database name MusicStore and add aTable named Album with the following fields

Album

AlbumID

Name

Genre

Year

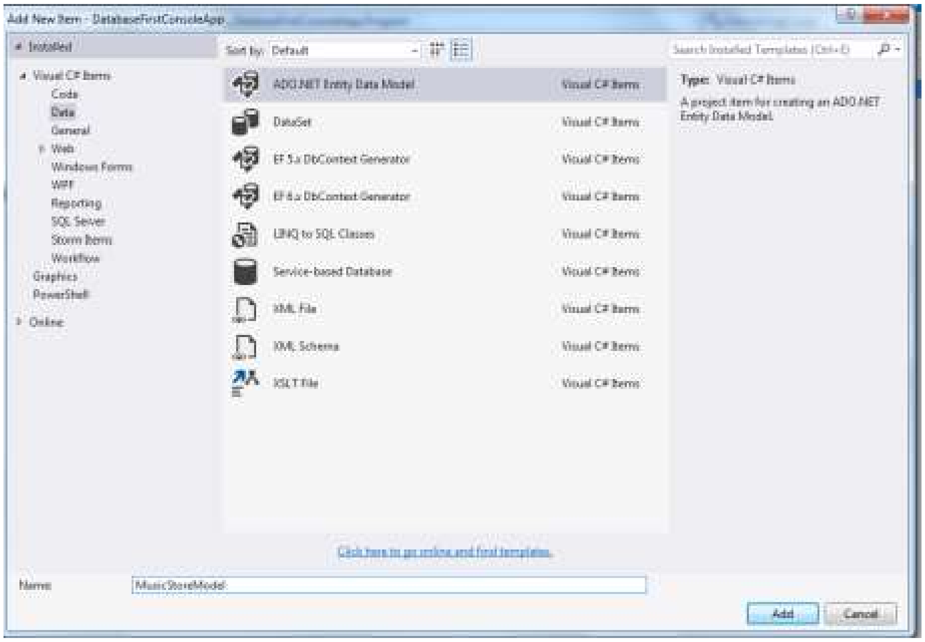
Price

Add some dummy record into the table.

Now create a console application name DatabaseFirstConsoleApp and add the entityframework as done in the previous example

Once the project in create and entity framework library is added . Now we have a Entity data Model to the project .

To add a Entity data Model to the Project in the solution explorer right click in the project Add -> New Item. Under the New Item dialog box select Ado.Net Entity Data Model and give name as MusicStoreModel and click on Add



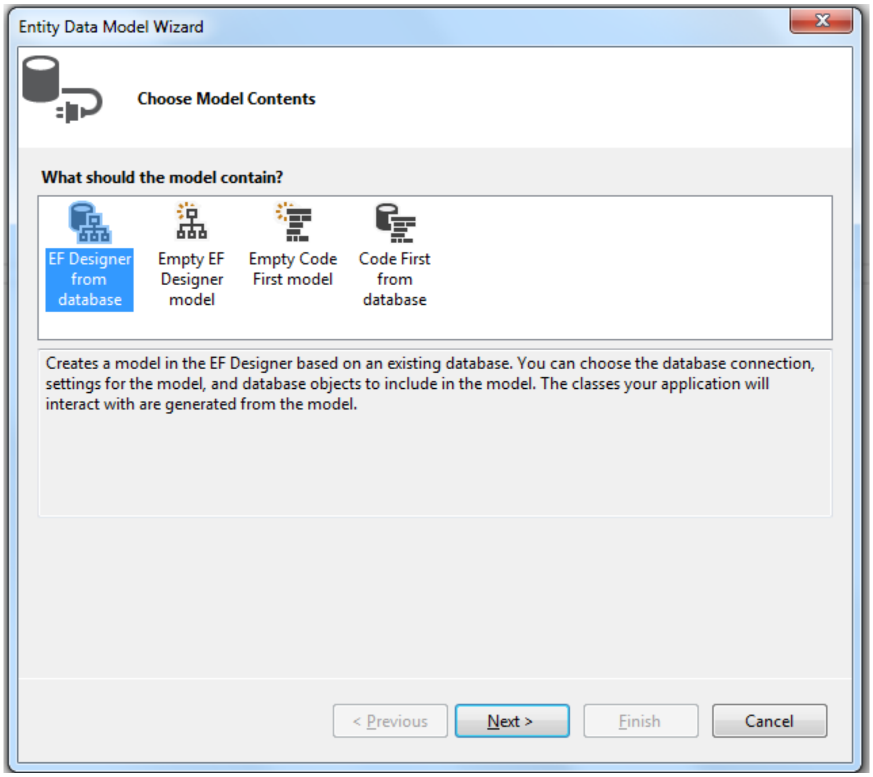
Type: Visual C# Items

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

Name: MusicStoreModel

As we add the Entity Data model to the project . Entity Data model wizard popup in which we have different option for initializing the entity data model

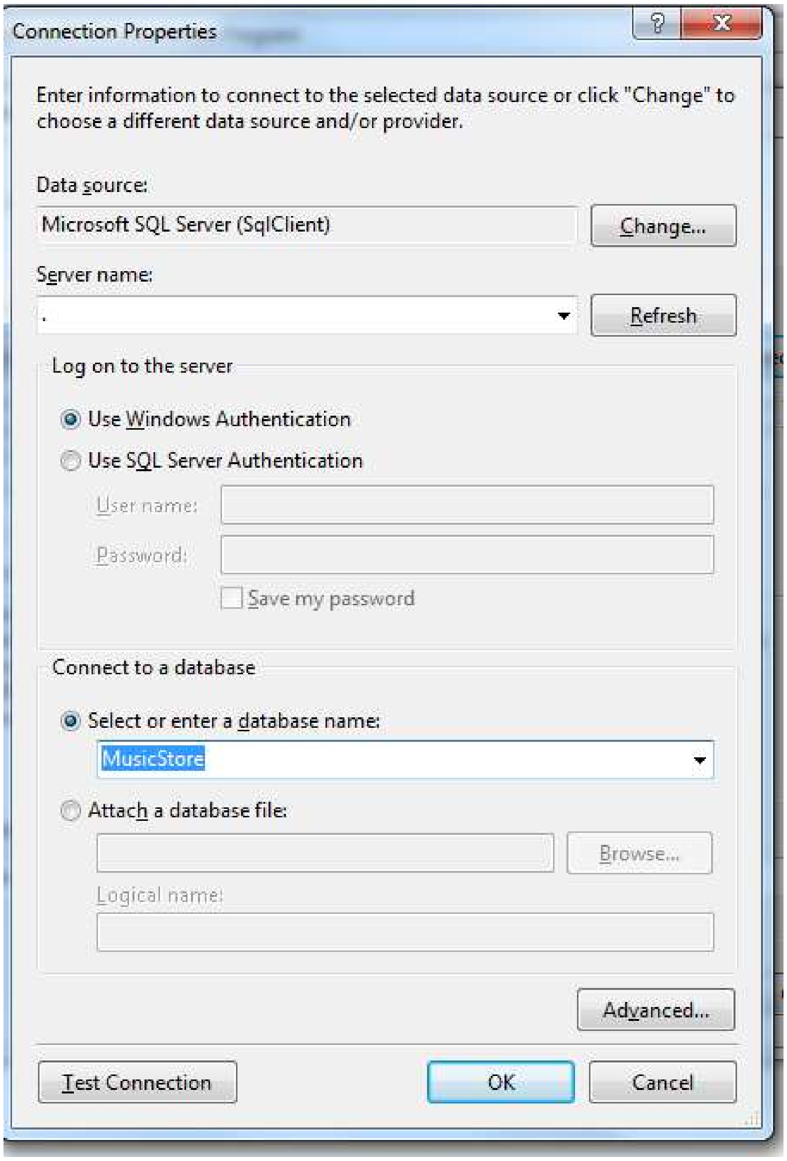
In that dialog box **select EF Designer from database** and click on Next



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.

Now on the next window we have select database for Model creation so now click on New Connection button in Choose Your Data Connection

In the Connection properties dialog box provide the Database server name and select the database you want to use. Click on Test Connection to test the connection and then click on OK



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: .

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

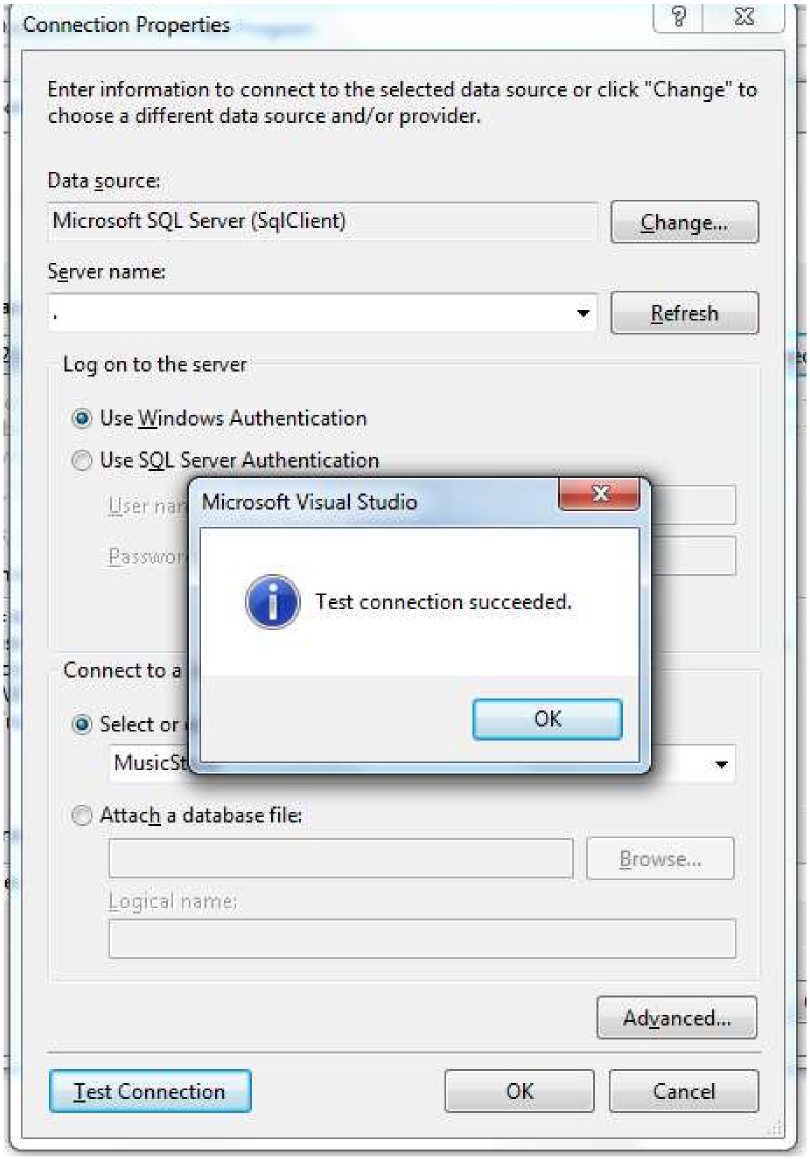
□ Save my password

Connect to a database

● Select or enter a database name: MusicStore

○ Attach a database file:

Logical name:



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: .

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

Connect to a

● Select or

MusicSt

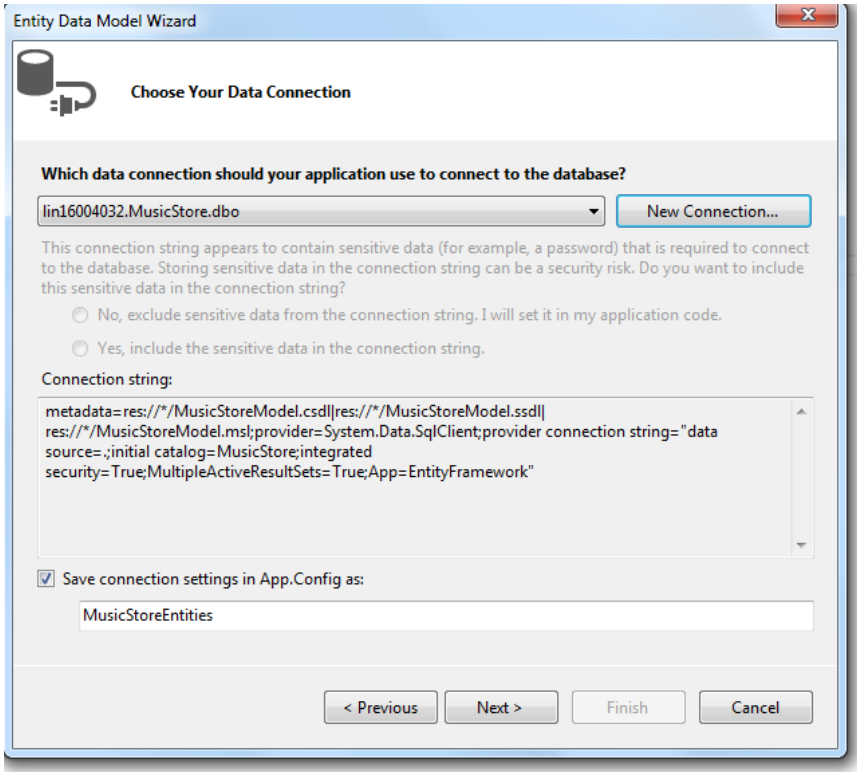
○ Attach a database file:

Microsoft Visual Studio

Test connection succeeded.

Once the connection test is passed then click on OK

Now we can see the new connection string which we have created and a option to save the connection string in App.config or web.config file



Entity Data Model Wizard

Choose Your Data Connection

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

lin16004032.MusicStore.dbo

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.

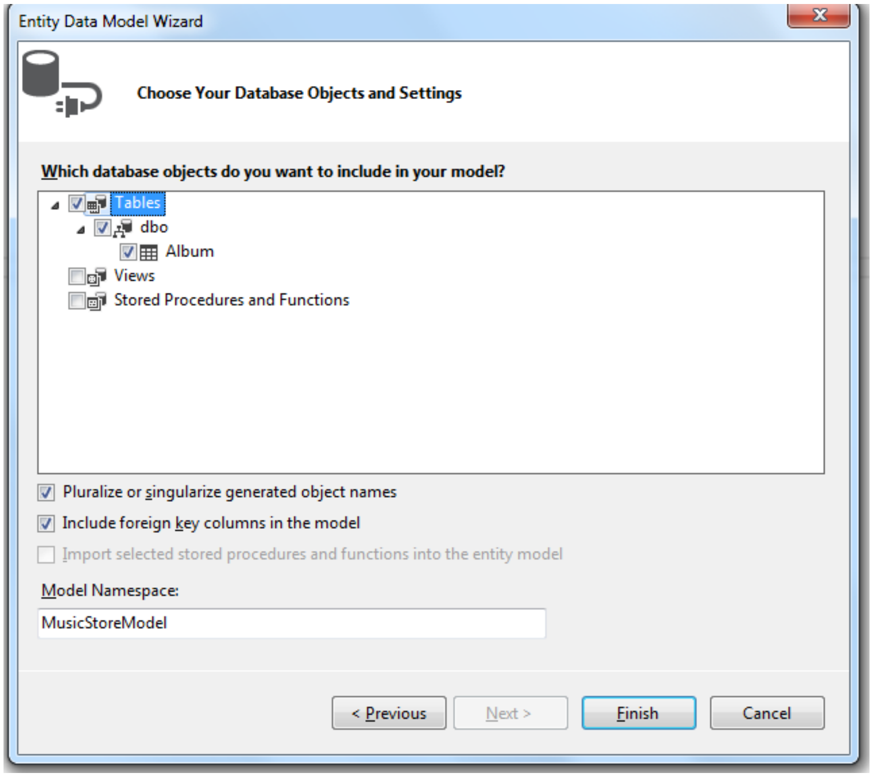
Connection string:

```
metadata=res://*/MusicStoreModel.csdl|res://*/MusicStoreModel.ssdl|
res://*/MusicStoreModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=.;initial catalog=MusicStore;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in App.Config as:

MusicStoreEntities

Now Click on Next and Choose your Database Object and Settings option will be prompted . In that we have select all the database object which we need to add to our Model.



Entity Data Model Wizard

Choose Your Database Objects and Settings

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

- ☑ Tables
  - ☑ dbo
    - ☑ Album
- □ Views
- □ Stored Procedures and Functions

☑ Pluralize or singularize generated object names

☑ Include foreign key columns in the model

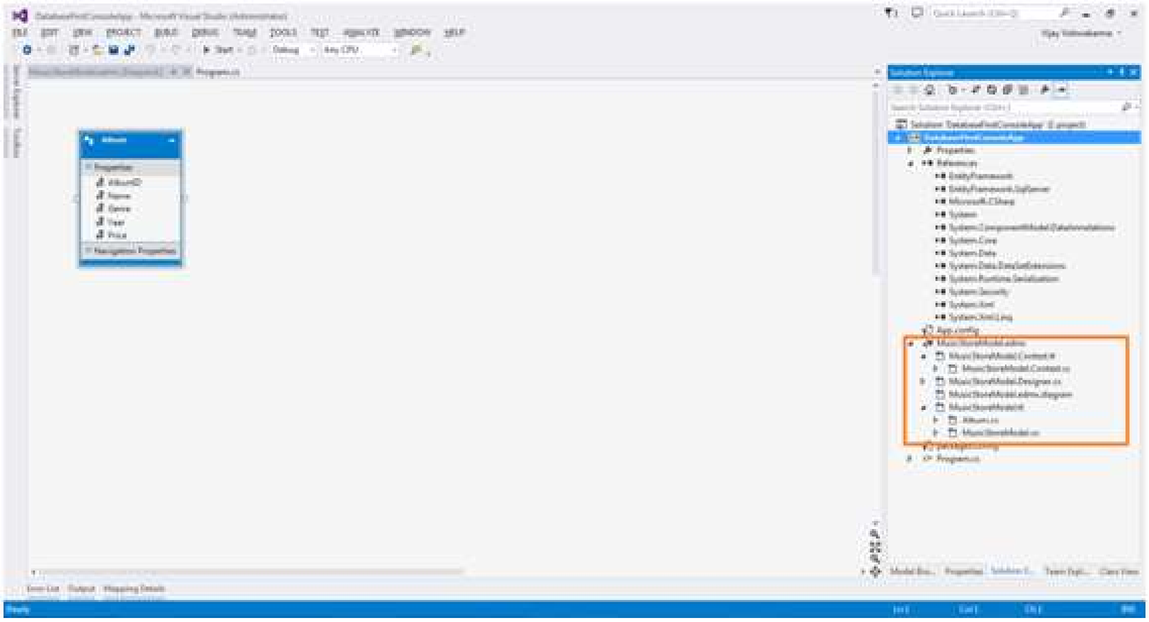
□ Import selected stored procedures and functions into the entity model

Model Namespace:

MusicStoreModel

We have select the Album table as we have only one table in the database

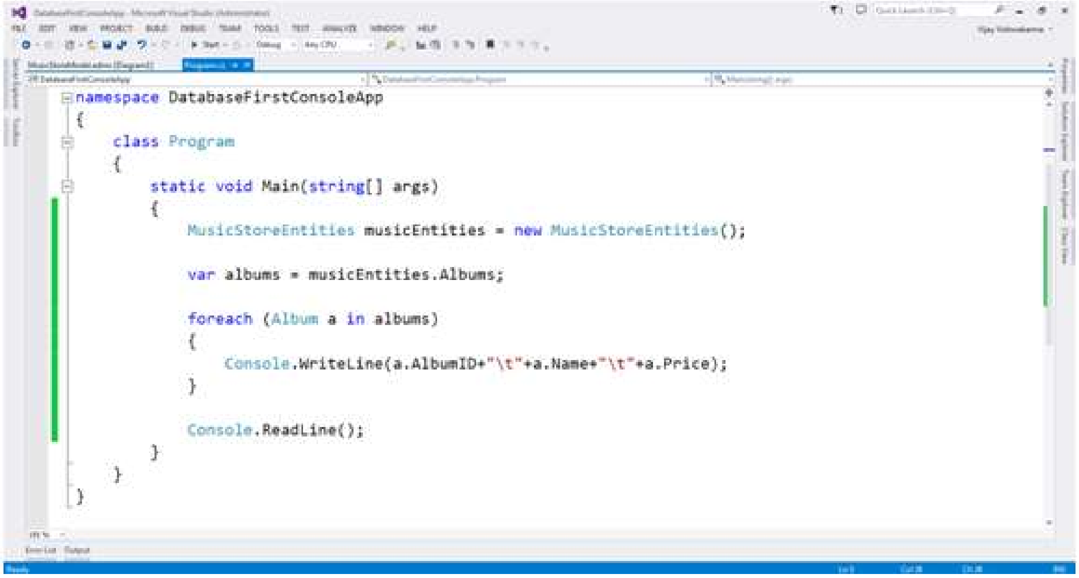
Now click on Finish this will add the EDM to the project and create all the required code.



In the above image we can see the model name MusicStoreModel.edmx containing Album Entity and the highlighted region show files generated for MusicStoreModel.edmx

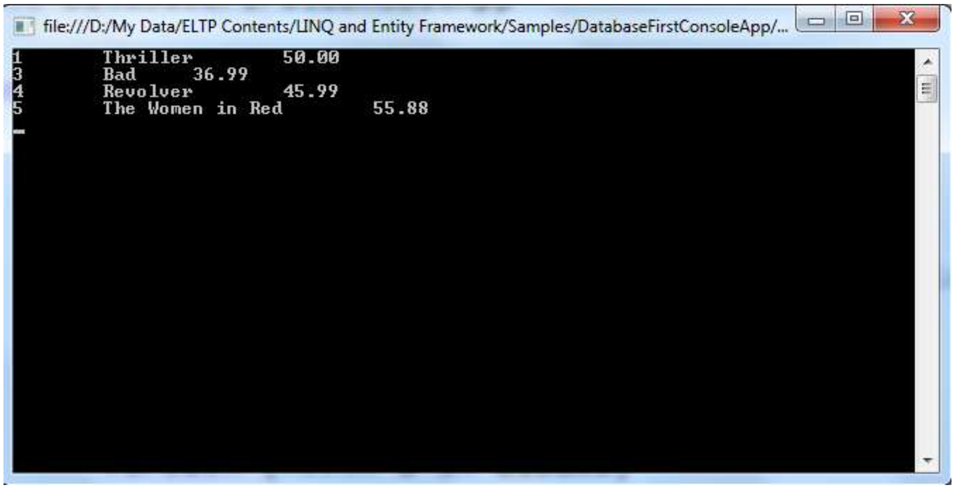
Now as the model is created we can write code to interact with database and perform read/write operations.

To display details of the Albums write down the following code in Program.cs File



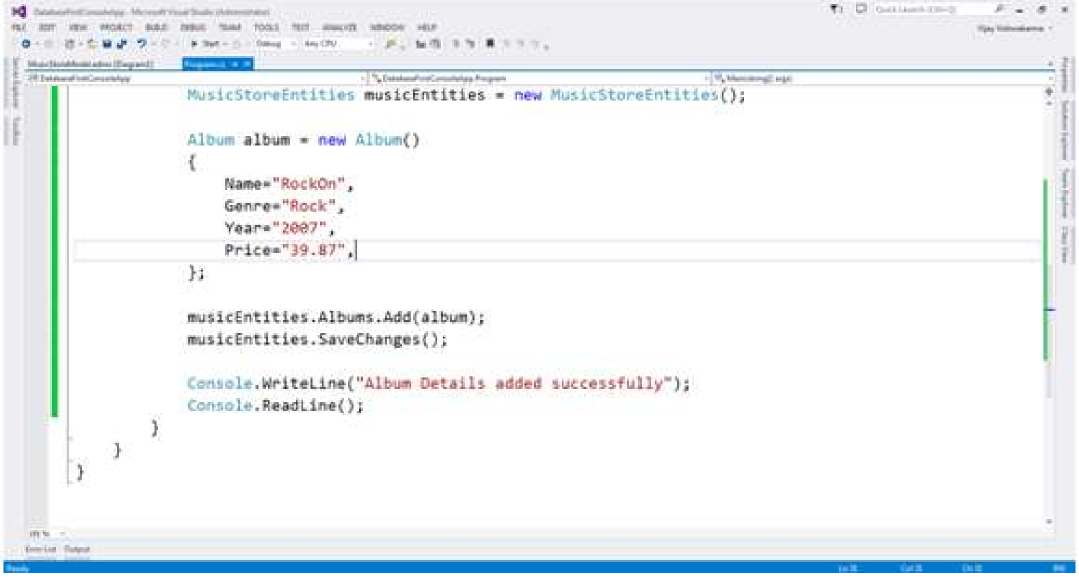
```
namespace DatabaseFirstConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            MusicStoreEntities musicEntities = new MusicStoreEntities();

            var albums = musicEntities.Albums;

            foreach (Album a in albums)
            {
                Console.WriteLine(a.AlbumID+"\t"+a.Name+"\t"+a.Price);
            }

            Console.ReadLine();
        }
    }
}
```

Output:-


file:///D:/My Data/ELTP Contents/LINQ and Entity Framework/Samples/DatabaseFirstConsoleApp/...

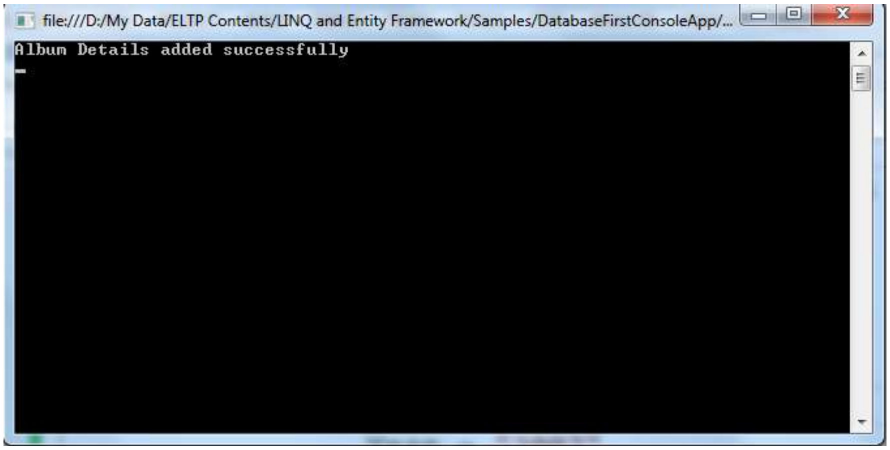
```
1     Thriller    50.00
3     Bad   36.99
4     Revolver    45.99
5     The Women in Red      55.88
```

To add an Album into the database table write the following code in Program.cs File

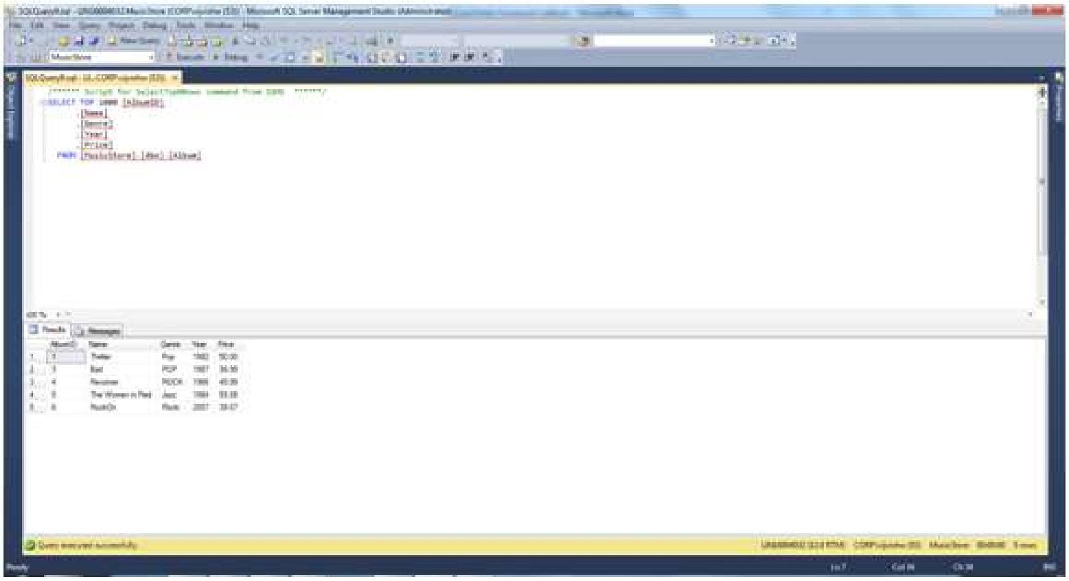


```
            MusicStoreEntities musicEntities = new MusicStoreEntities();

            Album album = new Album()
            {
                Name="RockOn",
                Genre="Rock",
                Year="2007",
                Price="39.87",
            };

            musicEntities.Albums.Add(album);
            musicEntities.SaveChanges();

            Console.WriteLine("Album Details added successfully");
            Console.ReadLine();
        }
    }
}
```

Output:-


file:///D:/My Data/ELTP Contents/LINQ and Entity Framework/Samples/DatabaseFirstConsoleApp/...

```
Album Details added successfully
```



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [AlbumID]
      ,[Name]
      ,[Genre]
      ,[Year]
      ,[Price]
  FROM [MusicStore].[dbo].[Album]
```

	AlbumID	Name	Genre	Year	Price
1	1	Thriller	Pop	1982	50.00
2	3	Bad	POP	1987	36.59
3	4	Revolver	ROCK	1966	45.99
4	5	The Women in Red	Jazz	1984	55.88
5	6	RockOn	Rock	2007	39.67

Query executed successfully.

Lab 4. Creating Entity Data Model using Model First Approach

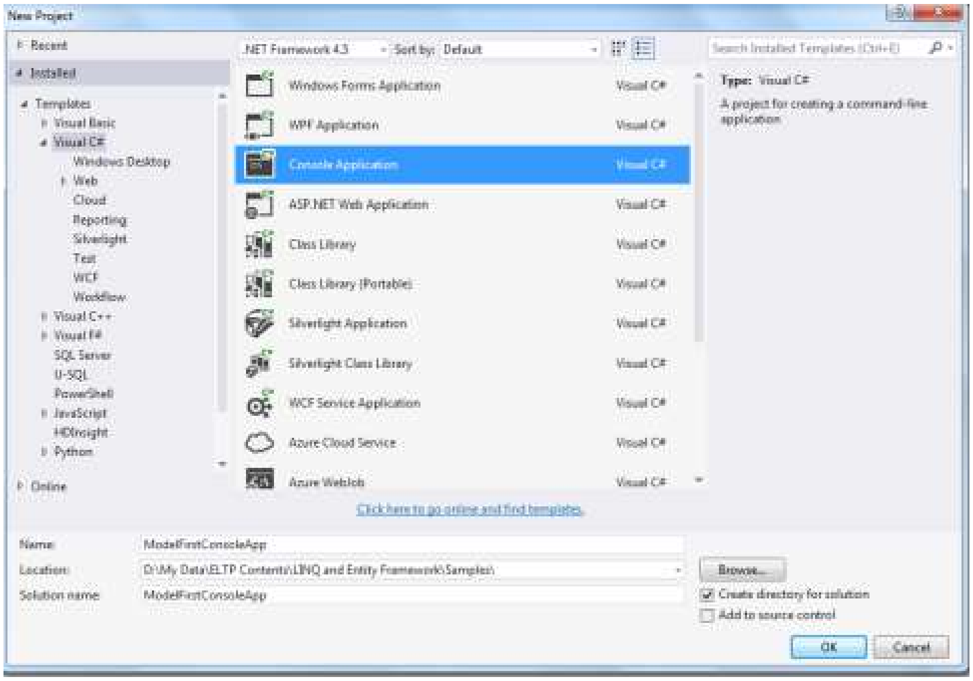
Goals	Understand the process of Creating Entity Data Model using Model First Approach and Creating Complex Type Learn to use of Model First Approach to create database tables
Time	60 minutes

Model First Approach :-

Model First allows you to create a new model using the Entity Framework Designer and then generate a database schema from the model. The model is stored in an EDMX file and can be viewed and edited in the Entity Framework Designer.

Solution:-

Open Visual Studio and create a console application named ModelFirstConsoleApp. After creating the project add Entityframework to the project using Nuget Package Manager


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows Desktop
Web
Cloud
Reporting
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
U-SQL
PowerShell
JavaScript
HDInsight
Python
Online
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
WCF Service Application Visual C#
Azure Cloud Service Visual C#
Azure WebJob Visual C#
Click here to go online and find templates.
Search Installed Templates (Ctrl+E)
Browse...
OK
Cancel

.NET Framework 4.5 Sort by: Default

Type: Visual C#

A project for creating a command-line application

Name: ModelFirstConsoleApp

Location: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\

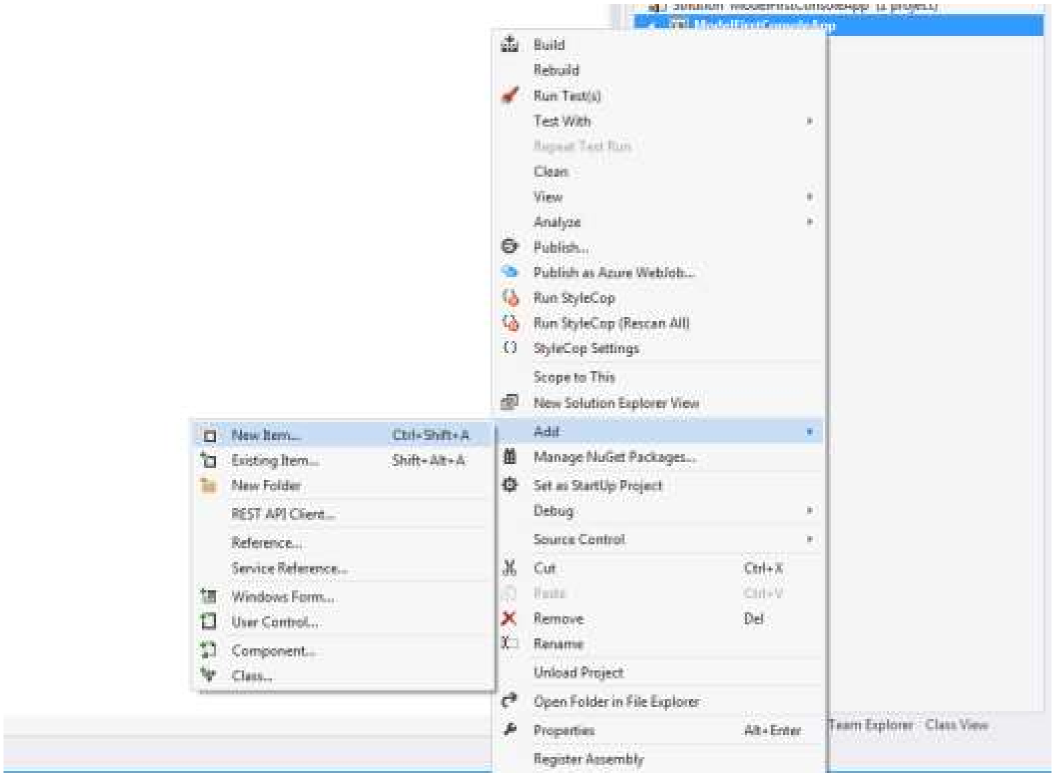
Solution name: ModelFirstConsoleApp

☑ Create directory for solution

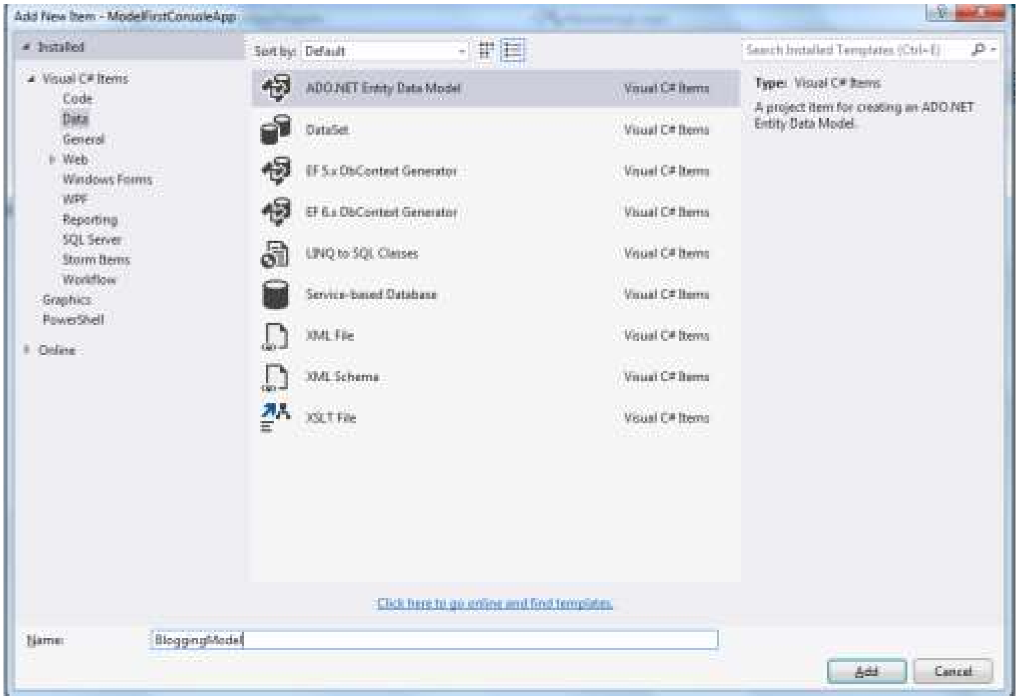
□ Add to source control

After creating the project and adding Entityframework to the project .Now we have a add a model to the project..

For adding a model to the project Right Click on the Project in Solution Explorer □Add□New Item



Now in the add new item dialog box select ADO.Net Entity Data Model and named it as BloggingModel and click on ADD

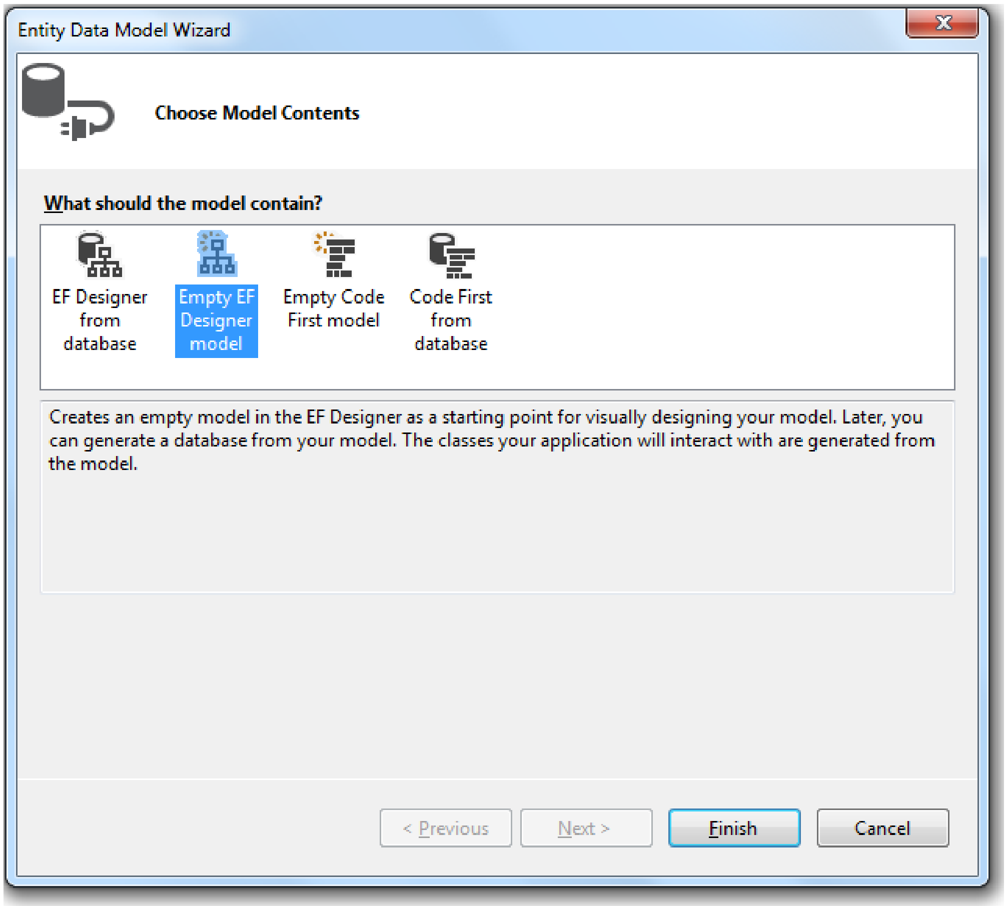

Add New Item - ModelFirstConsoleApp
Installed
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
SQL Server
Storm Items
Workflow
Graphics
PowerShell
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
ADO.NET Entity Data Model Visual C# Items
DataSet Visual C# Items
EF 5.x DbContext Generator Visual C# Items
EF 6.x DbContext Generator Visual C# Items
LINQ to SQL Classes Visual C# Items
Service-based Database Visual C# Items
XML File Visual C# Items
XML Schema Visual C# Items
XSLT File Visual C# Items
Click here to go online and find templates.
Add
Cancel

Type: Visual C# Items

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

Name: BloggingModel

In the Entity Data Model Wizard select Empty EF Designer Model and click on Finish.



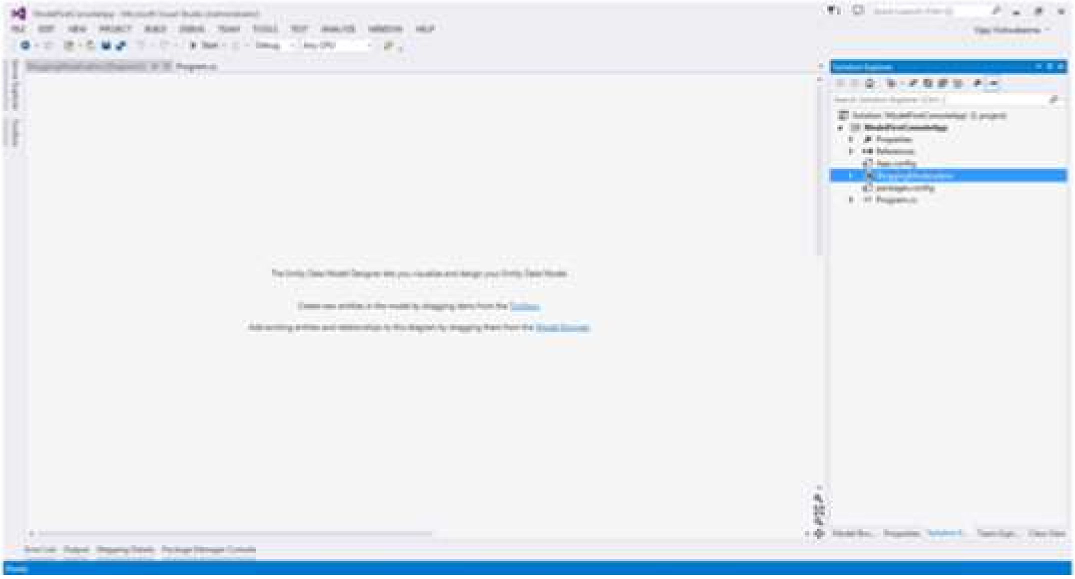
Entity Data Model Wizard

Choose Model Contents

What should the model contain?

Creates an empty model in the EF Designer as a starting point for visually designing your model. Later, you can generate a database from your model. The classes your application will interact with are generated from the model.

This will open up an empty EDMX on which we create Model from Scratch



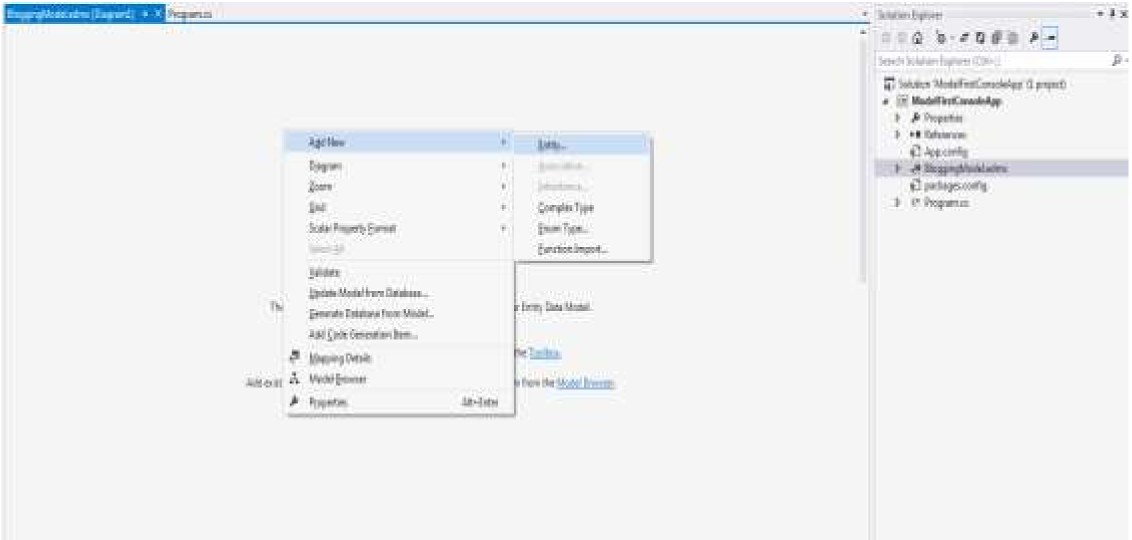
The Entity Data Model Designer lets you visualize and design your Entity Data Model.

Create new entities in the model by dragging items from the Toolbox.

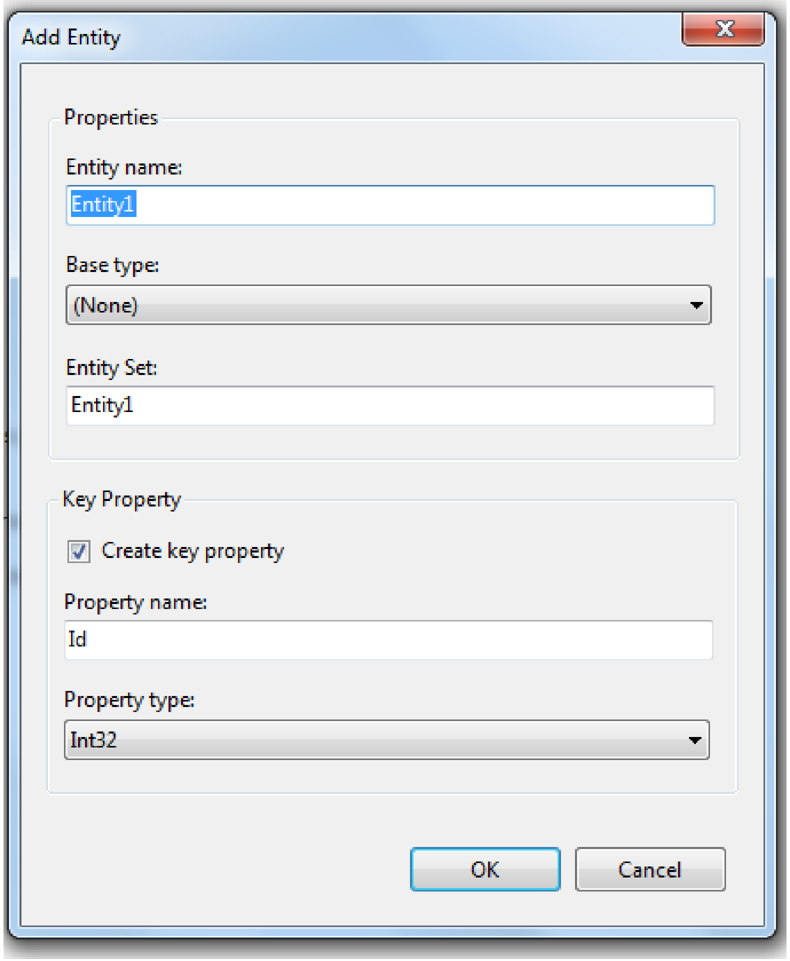
Add existing entities and relationships to this diagram by dragging them from the Model Browser.

Now we have to add entity in this model and create a database form it.

To add a Entity , Right click on the open edmx file and select Add New -> Entity



After you select you will be prompted with the following Dialog for creating a new entity



Properties

Entity name: Entity1

Base type: (None)

Entity Set: Entity1

Key Property

☑ Create key property

Property name: Id

Property type: Int32

The Dialog Display information like

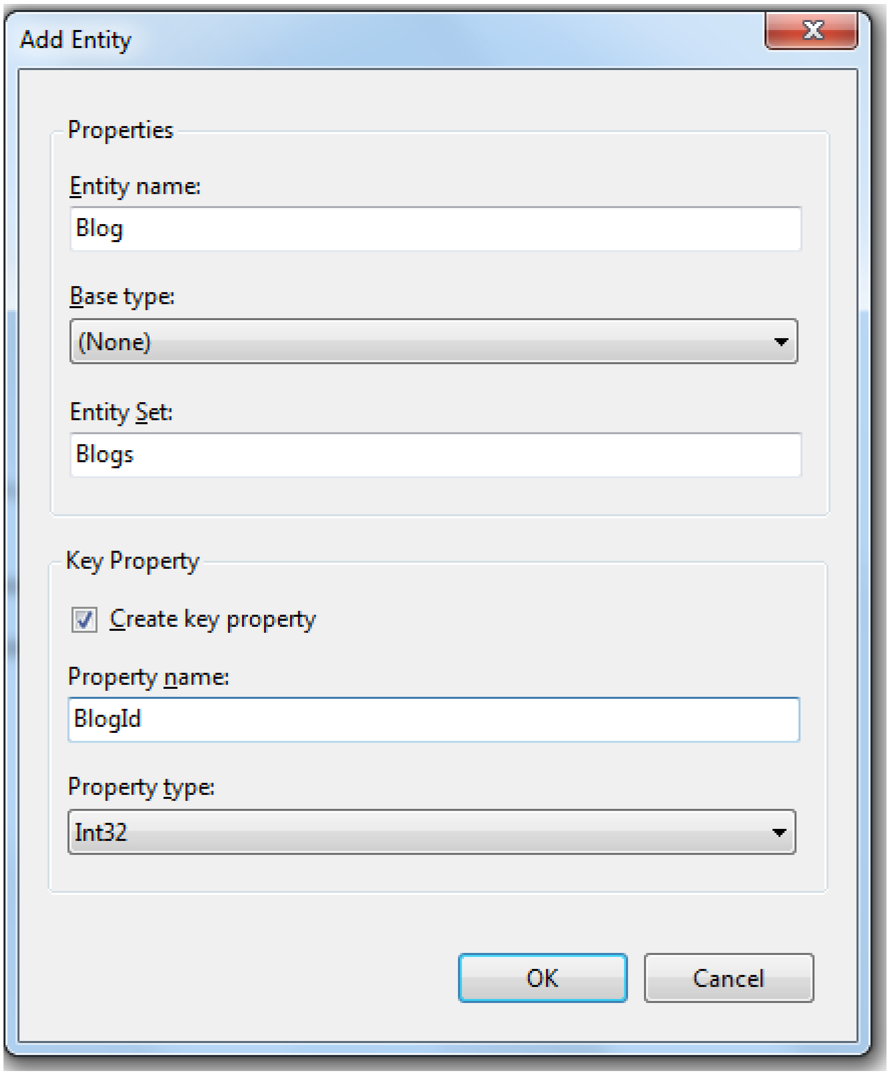
Entity Name :- This will the name of Entity

Base Type :- Shows the base type of the entity

EntitySet :- Show the name of the Entity Set

It also show the option for creating Key Property where we have to specify Key Property Name and its Data Type

For our example we will Name the entity as Blog and Entity Set as Blogs .Entity Set is automatically Pluralize as we add the name of the Entity. and change Key Property from Id to BlogId



Entity name: Blog

Base type: (None)

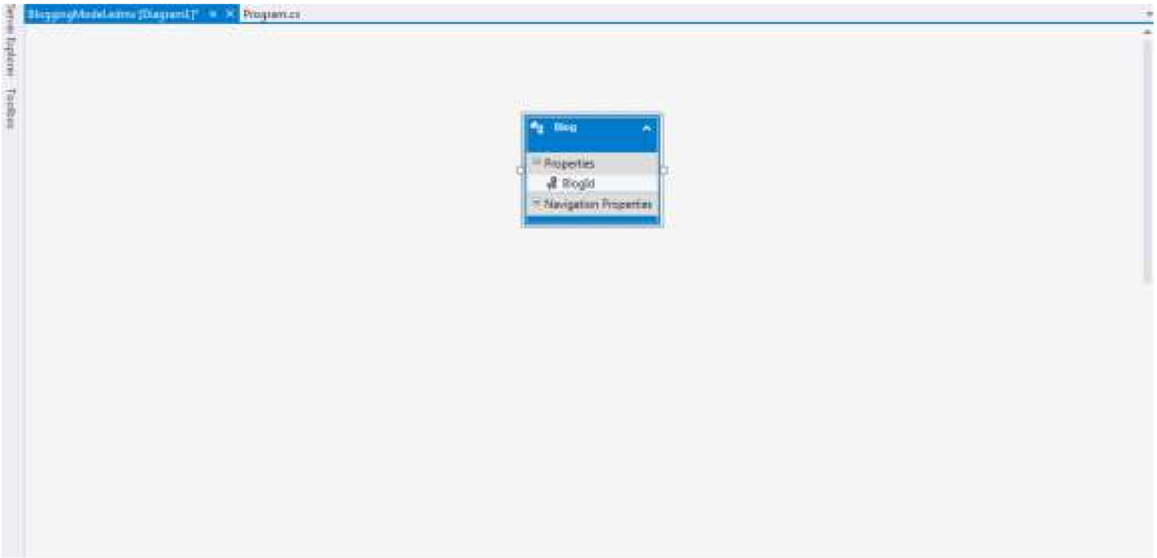
Entity Set: Blogs

☑ Create key property

Property name: BlogId

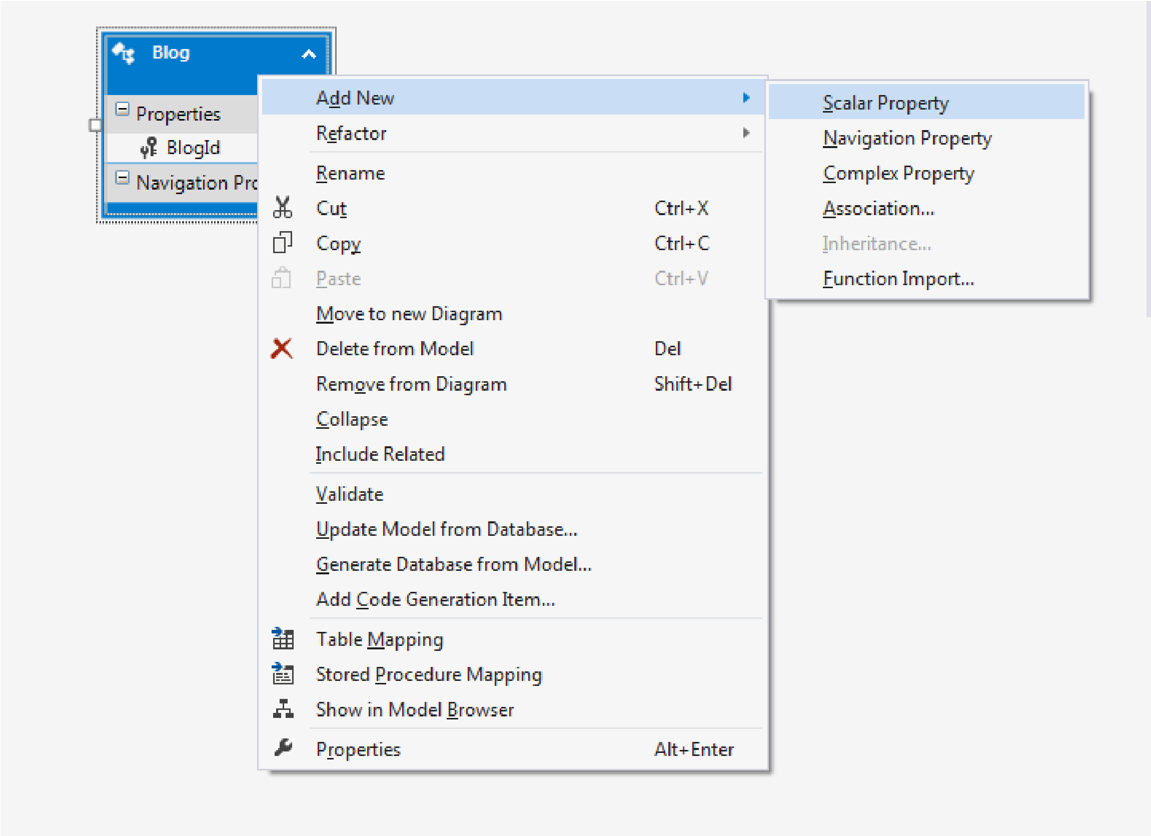
Property type: Int32

After changing all the details clik on OK .This will add the entity to EDMX designer

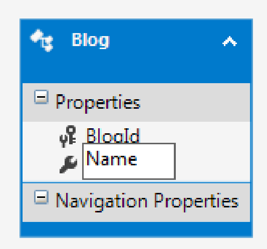


Now we have add Field to this entity for storing other information of Blog.

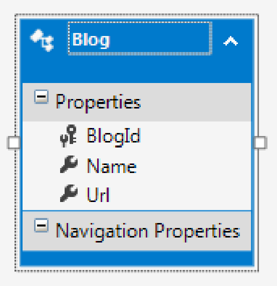
To add a Field or property to the Entity right on the Entity and select Add New -> Scalar Property



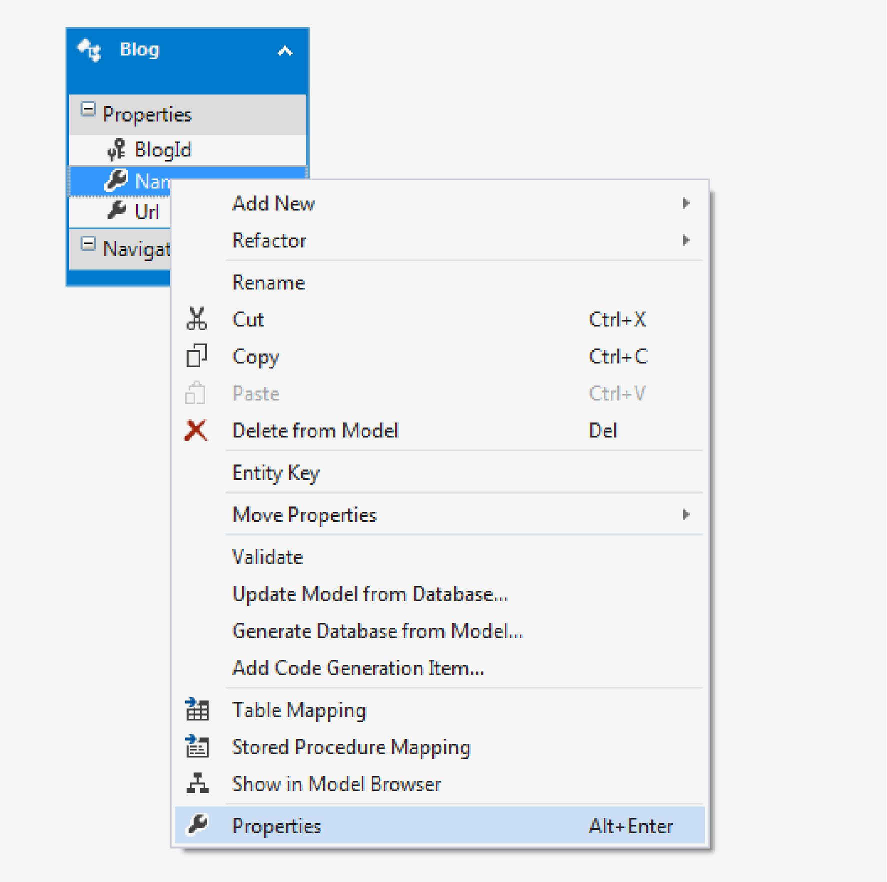
After selecting the Scalar Property , name it as Name

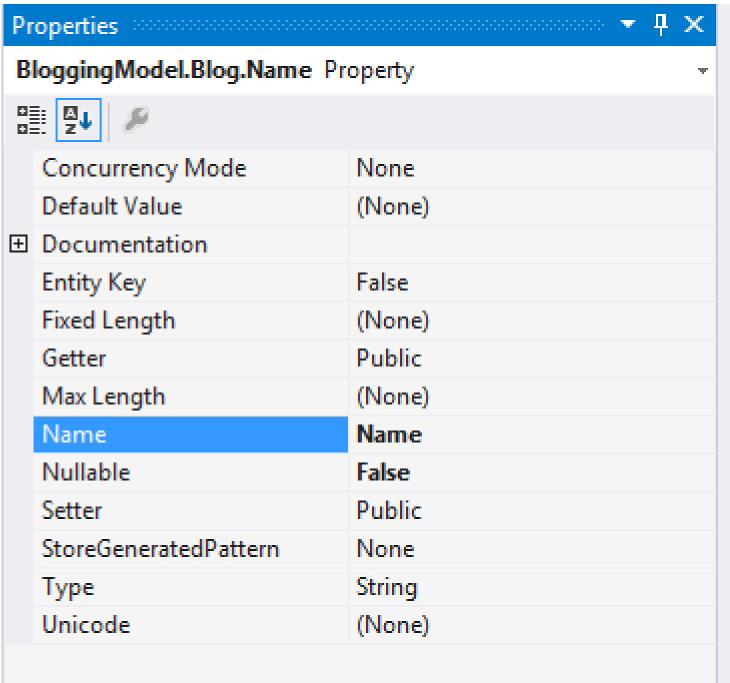


Similarly add another property named URL to the Entity. After adding the URL property the Entity Will look as follows



We can view the properties of each Property add the Entity by just right clicking and selecting the properties option





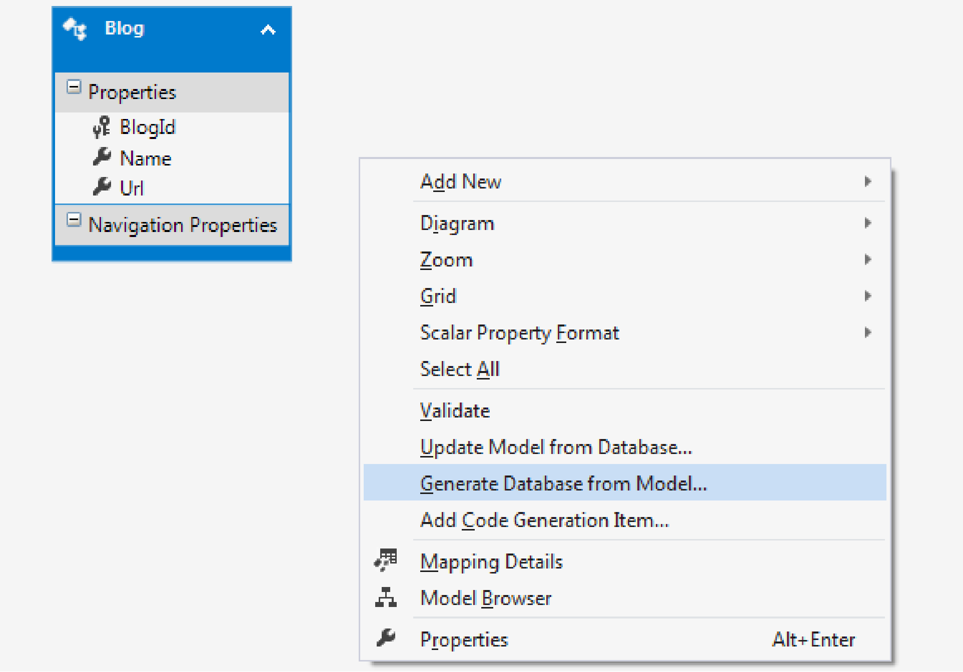
Properties

BloggingModel.Blog.Name Property

Concurrency Mode	None
Default Value	(None)
⊞ Documentation	
Entity Key	False
Fixed Length	(None)
Getter	Public
Max Length	(None)
Name	**Name**
Nullable	**False**
Setter	Public
StoreGeneratedPattern	None
Type	String
Unicode	(None)

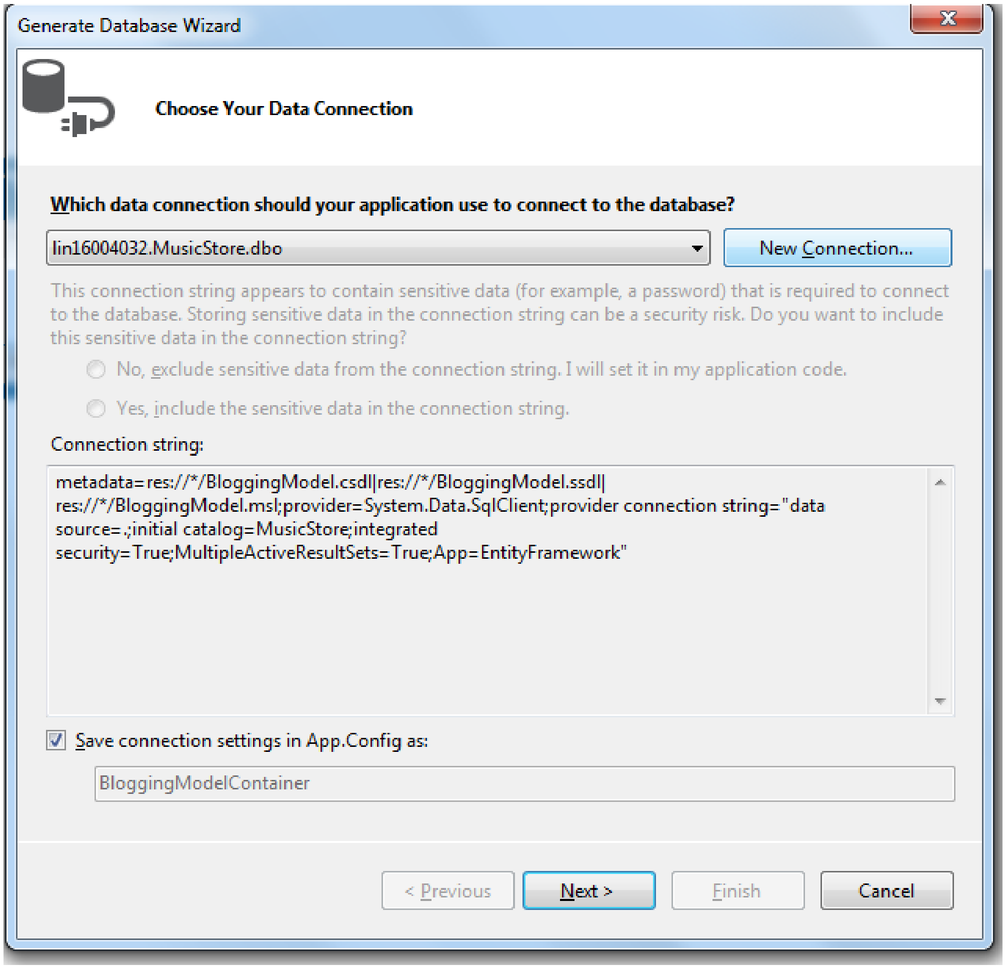
We can see above the properties show a lot of detail about the Name scalar property add to the entity.

Once the Entity has been created , now we have to create Database from the Model . To create the database from the model right click on the EDMX designer and select Generate Database from Model



After you click on the Generate Database from Model , Generate Database Wizard will be prompted .

In the Generate Database Wizard click on the New Connection to create a new connection .



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

lin16004032.MusicStore.dbo

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.

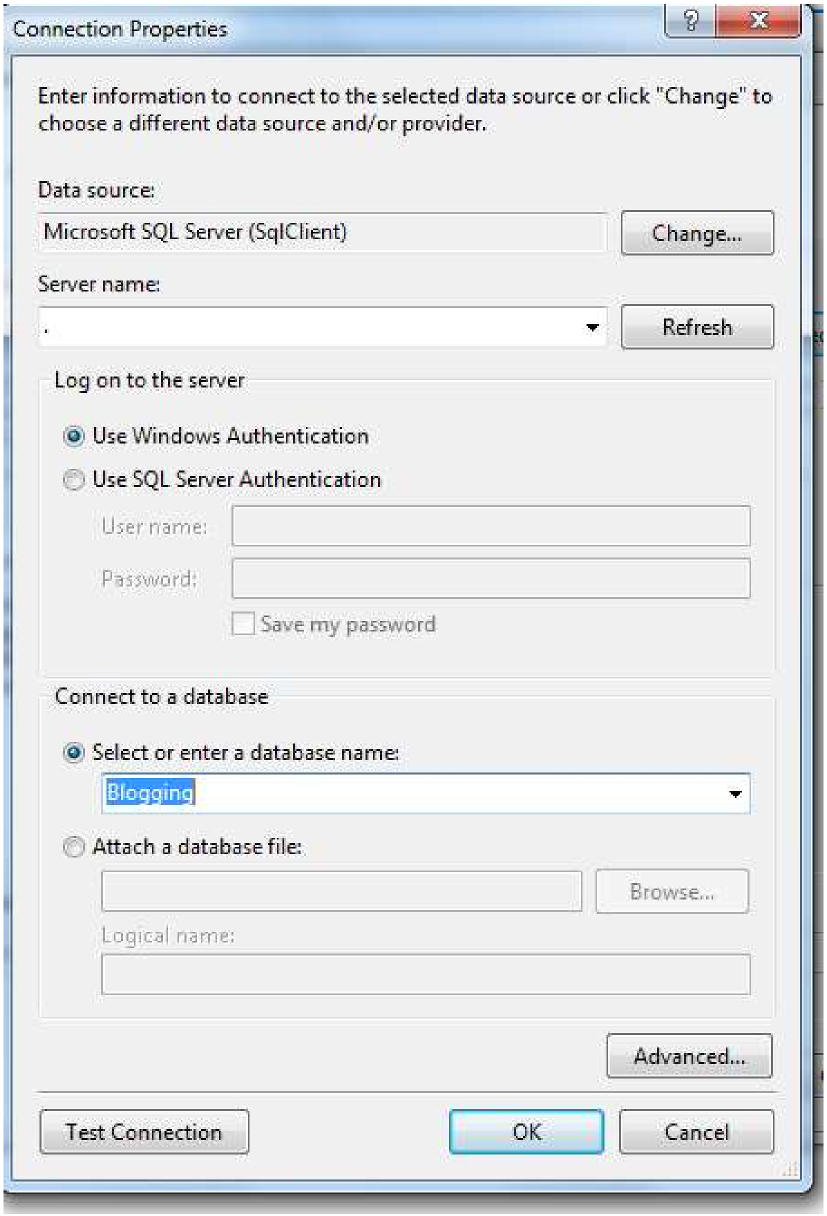
○ Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/BloggingModel.csdl|res://*/BloggingModel.ssdl|
res://*/BloggingModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=.;initial catalog=MusicStore;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in App.Config as:

BloggingModelContainer



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient)

Server name:
.

Log on to the server

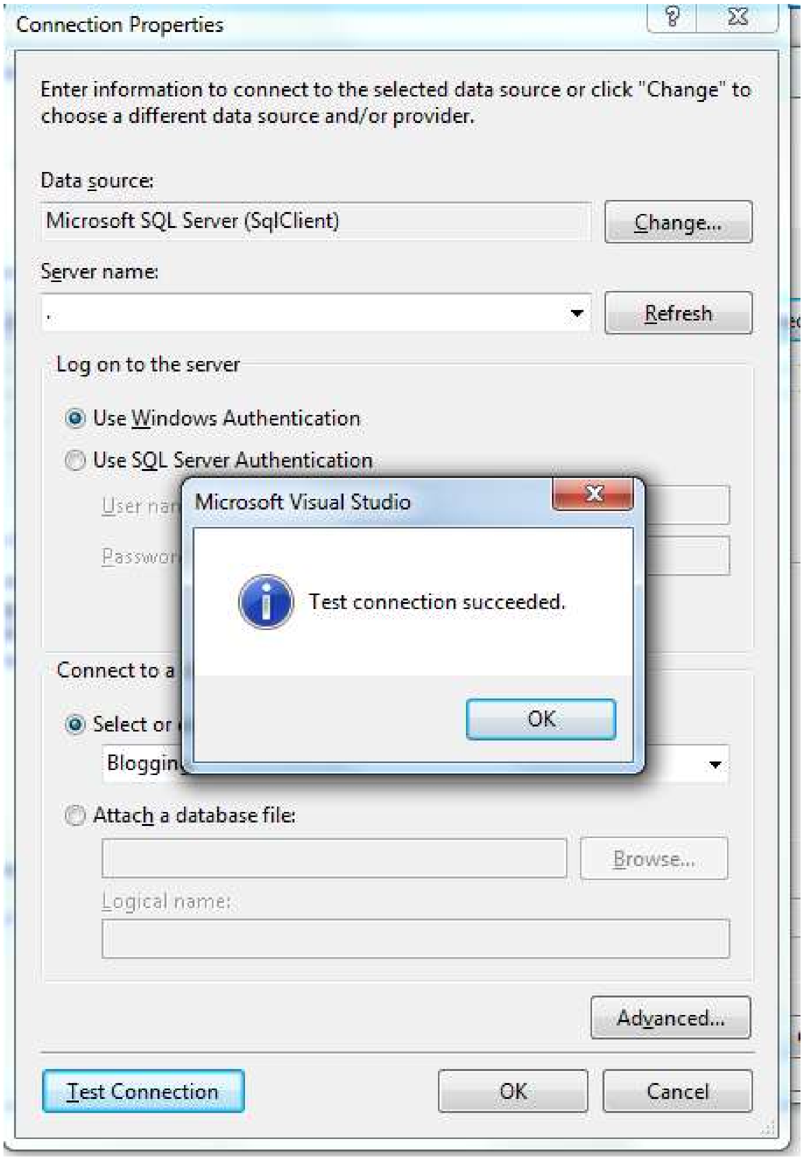
● Use Windows Authentication
○ Use SQL Server Authentication

User name:
Password:
□ Save my password

Connect to a database

● Select or enter a database name:
Blogging
○ Attach a database file:

Logical name:



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: .

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

Connect to a

● Select or

Bloggin

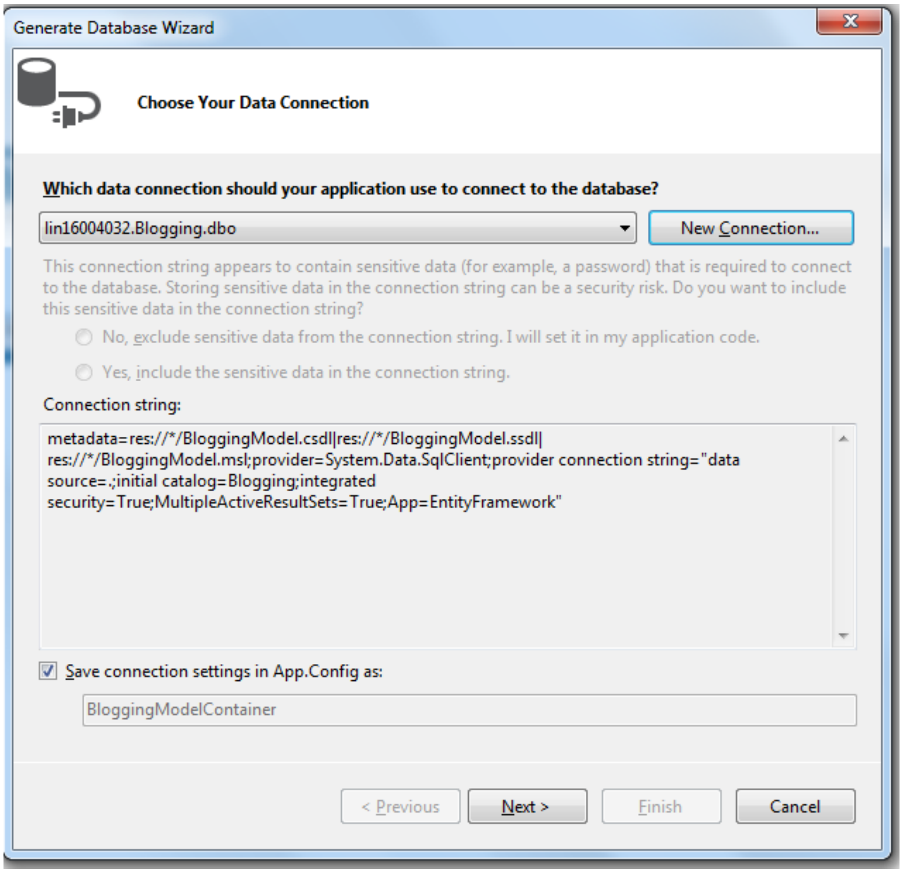
○ Attach a database file:

Microsoft Visual Studio

Test connection succeeded.

After testing the connection click on OK .

Now we have a new connection in the Generate Database Wizard



Generate Database Wizard

Choose Your Data Connection

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

lin16004032.Blogging.dbo

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.

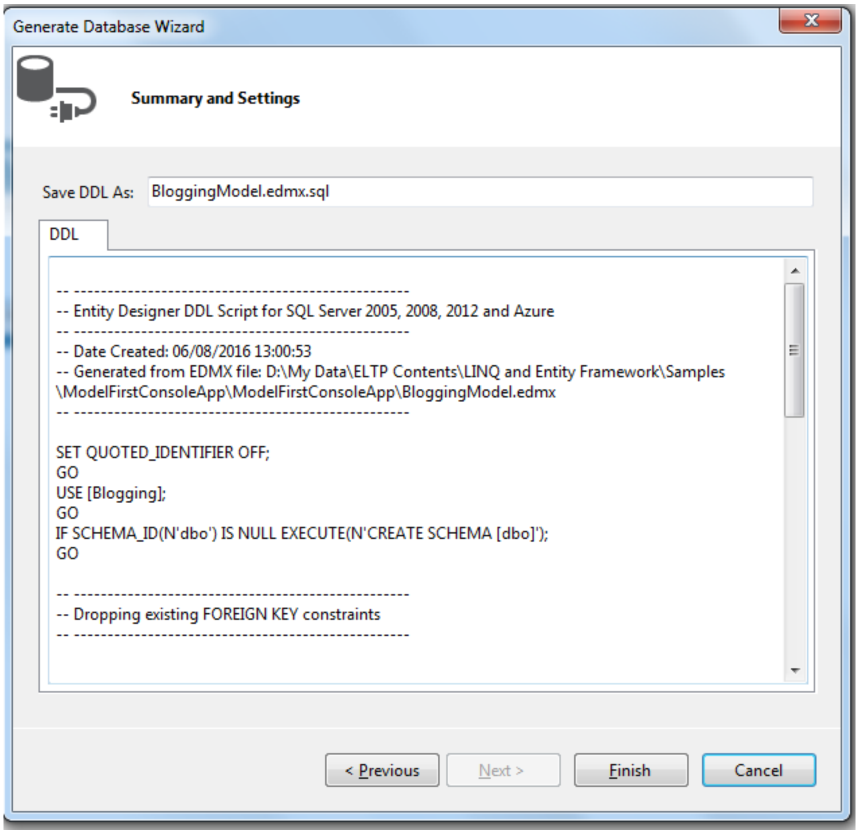
Connection string:

```
metadata=res://*/BloggingModel.csdl|res://*/BloggingModel.ssdl|
res://*/BloggingModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=.;initial catalog=Blogging;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in App.Config as:

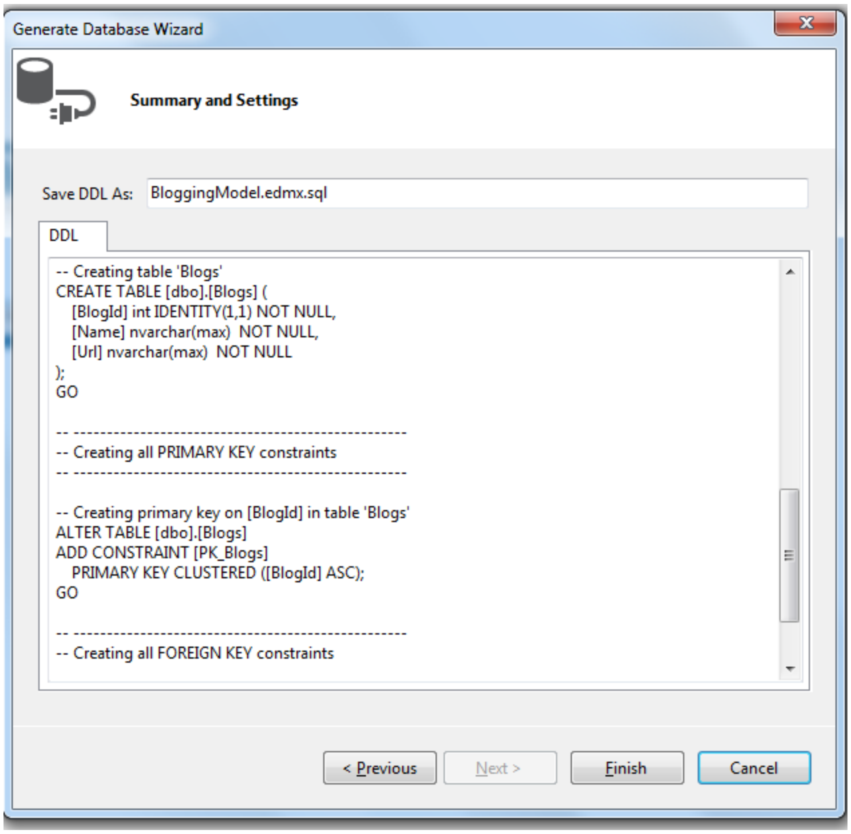
BloggingModelContainer

Now click on Next , DDL query for generating the Database and table will be auto generated.



Save DDL As: BloggingModel.edmx.sql

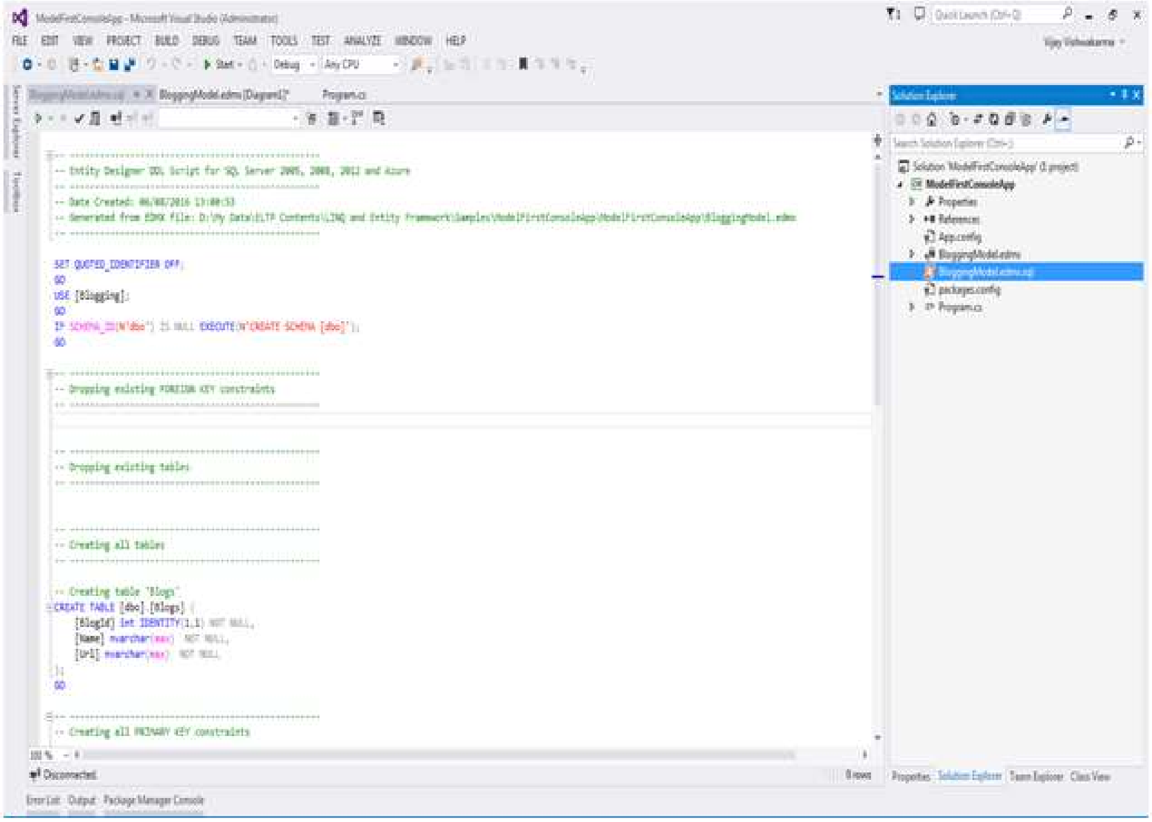
```
-- --------------------------------------------------
-- Entity Designer DDL Script for SQL Server 2005, 2008, 2012 and Azure
-- --------------------------------------------------
-- Date Created: 06/08/2016 13:00:53
-- Generated from EDMX file: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples
\ModelFirstConsoleApp\ModelFirstConsoleApp\BloggingModel.edmx
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
GO
USE [Blogging];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

-- --------------------------------------------------
-- Dropping existing FOREIGN KEY constraints
-- --------------------------------------------------
```



Save DDL As: BloggingModel.edmx.sql

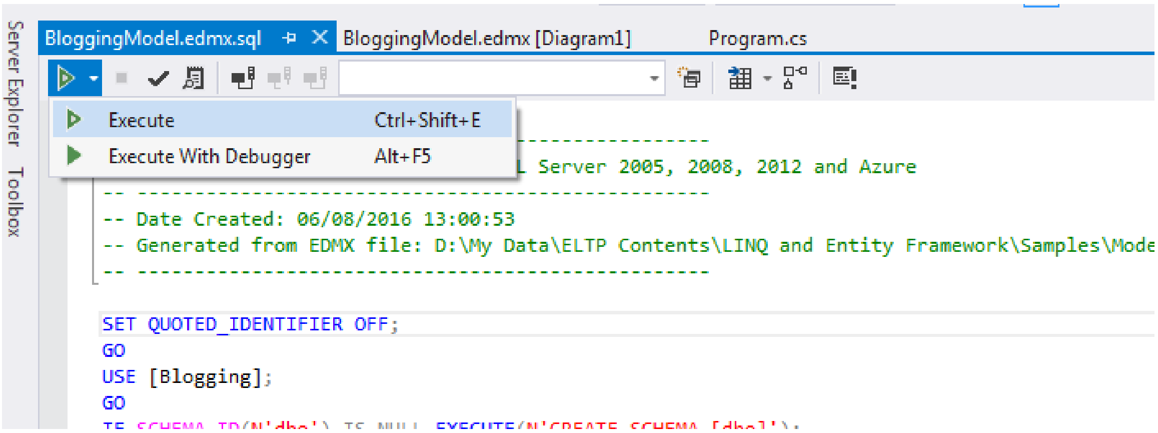
```
-- Creating table 'Blogs'
CREATE TABLE [dbo].[Blogs] (
    [BlogId] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(max)  NOT NULL,
    [Url] nvarchar(max)  NOT NULL
);
GO

-- --------------------------------------------------
-- Creating all PRIMARY KEY constraints
-- --------------------------------------------------

-- Creating primary key on [BlogId] in table 'Blogs'
ALTER TABLE [dbo].[Blogs]
ADD CONSTRAINT [PK_Blogs]
    PRIMARY KEY CLUSTERED ([BlogId] ASC);
GO

-- --------------------------------------------------
-- Creating all FOREIGN KEY constraints
```

Now click on finish , a file named BloggingModel.edmx.sql will be created which will contain sql shown in the dialog for creating the database.


ModelFirstConsoleApp - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Start
Debug
Any CPU
BloggingModel.edmx.sql
BloggingModel.edmx [Diagram1]*
Program.cs
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ModelFirstConsoleApp' (1 project)
ModelFirstConsoleApp
Properties
References
App.config
BloggingModel.edmx
BloggingModel.edmx.sql
packages.config
Program.cs
100 %
Disconnected.
0 rows
Properties Solution Explorer Team Explorer Class View
Error List Output Package Manager Console

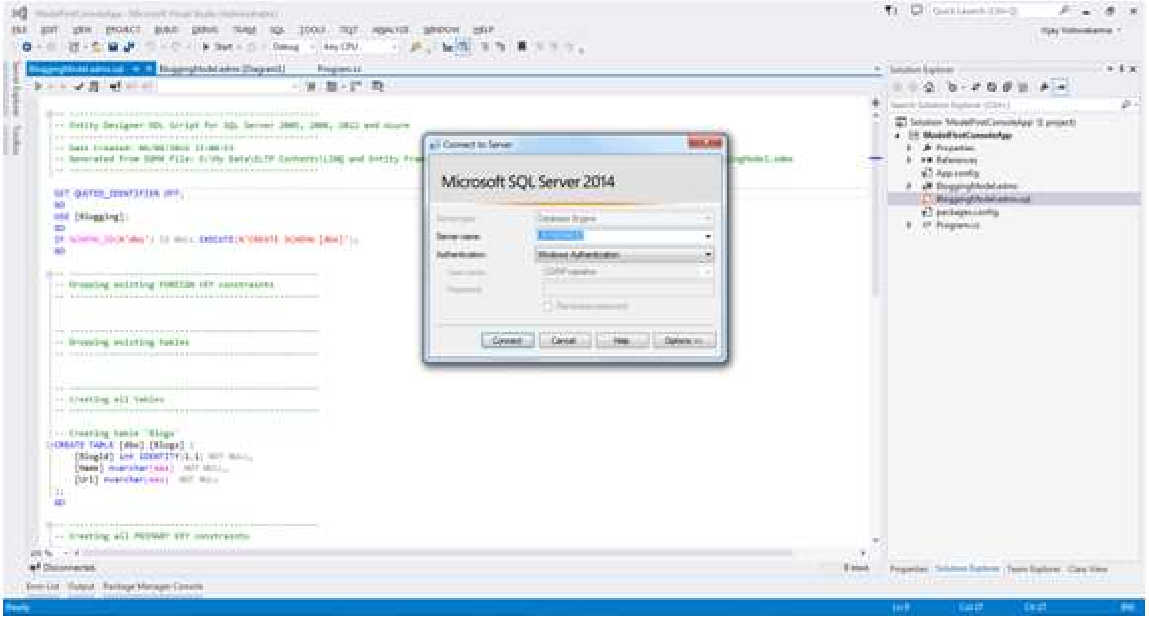
```
-- --------------------------------------------------
-- Entity Designer DDL Script for SQL Server 2005, 2008, 2012 and Azure
-- --------------------------------------------------
-- Date Created: 06/08/2016 13:00:53
-- Generated from EDMX file: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\ModelFirstConsoleApp\ModelFirstConsoleApp\BloggingModel.edmx
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
GO
USE [Blogging];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

-- --------------------------------------------------
-- Dropping existing FOREIGN KEY constraints
-- --------------------------------------------------


-- --------------------------------------------------
-- Dropping existing tables
-- --------------------------------------------------


-- --------------------------------------------------
-- Creating all tables
-- --------------------------------------------------

-- Creating table 'Blogs'
CREATE TABLE [dbo].[Blogs] (
    [BlogId] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(max)  NOT NULL,
    [Url] nvarchar(max)  NOT NULL
);
GO

-- --------------------------------------------------
-- Creating all PRIMARY KEY constraints
```

Now In the BloggingModel.edx.sql file select Execute and click on it



```
-- --------------------------------------------------
-- Entity Designer DDL Script for SQL Server 2005, 2008, 2012 and Azure
-- --------------------------------------------------
-- Date Created: 06/08/2016 13:00:53
-- Generated from EDMX file: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\Mode
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
GO
USE [Blogging];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
```

This will execute the query against the connected database instance if not connected you will be prompted to connect to sql server instance.



```
-- --------------------------------------------------
-- Entity Designer DDL Script for SQL Server 2005, 2008, 2012 and Azure
-- --------------------------------------------------
-- Date Created: [illegible]
-- Generated from EDMX file: E:\My Data\ELTP Contents\LINQ and Entity Fra
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
GO
USE [Blogging];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

-- --------------------------------------------------
-- Dropping existing FOREIGN KEY constraints
-- --------------------------------------------------


-- --------------------------------------------------
-- Dropping existing tables
-- --------------------------------------------------


-- --------------------------------------------------
-- Creating all tables
-- --------------------------------------------------

-- Creating table 'Blogs'
CREATE TABLE [dbo].[Blogs] (
    [BlogId] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(max)  NOT NULL,
    [Url] nvarchar(max)  NOT NULL
);
GO

-- --------------------------------------------------
-- Creating all PRIMARY KEY constraints
```

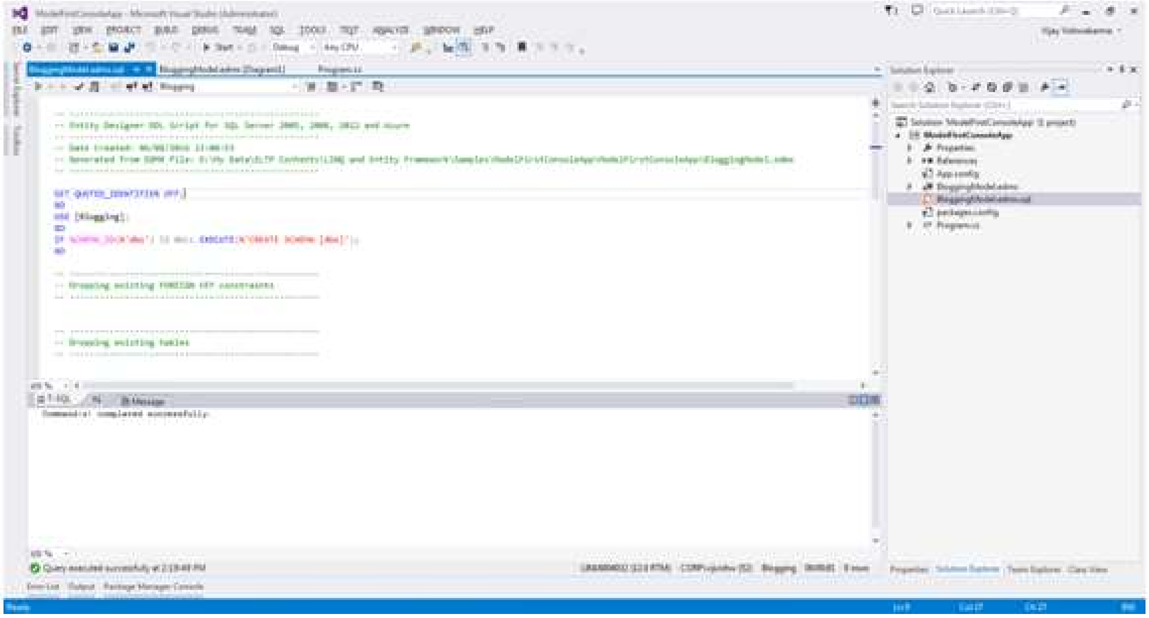
Connect to Server

Microsoft SQL Server 2014

Server type: Database Engine
Server name: [illegible]
Authentication: Windows Authentication

Connect | Cancel | Help | Options >>

Click on connect and query will executed

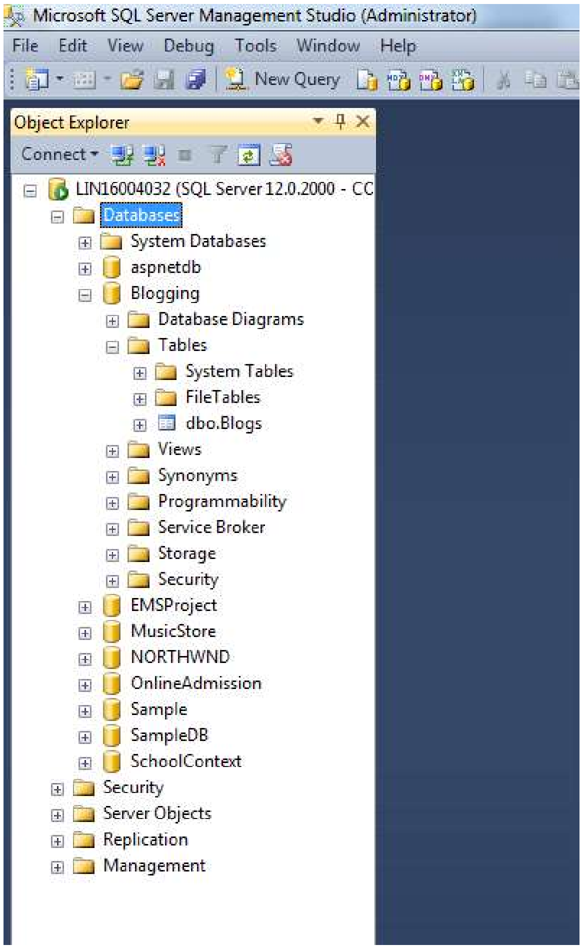


```
-- --------------------------------------------------
-- Entity Designer DDL Script for SQL Server 2005, 2008, 2012 and Azure
-- --------------------------------------------------
-- Date Created: [illegible]
-- Generated from EDMX file: E:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\ModelFirstConsoleApp\ModelFirstConsoleApp\BloggingModel.edmx
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
GO
USE [Blogging];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

-- --------------------------------------------------
-- Dropping existing FOREIGN KEY constraints
-- --------------------------------------------------


-- --------------------------------------------------
-- Dropping existing tables
-- --------------------------------------------------
```

```
Command(s) completed successfully.
```

Query executed successfully at 2:19:49 PM

Now in the Sql Server Management Studio we can the table which has been created



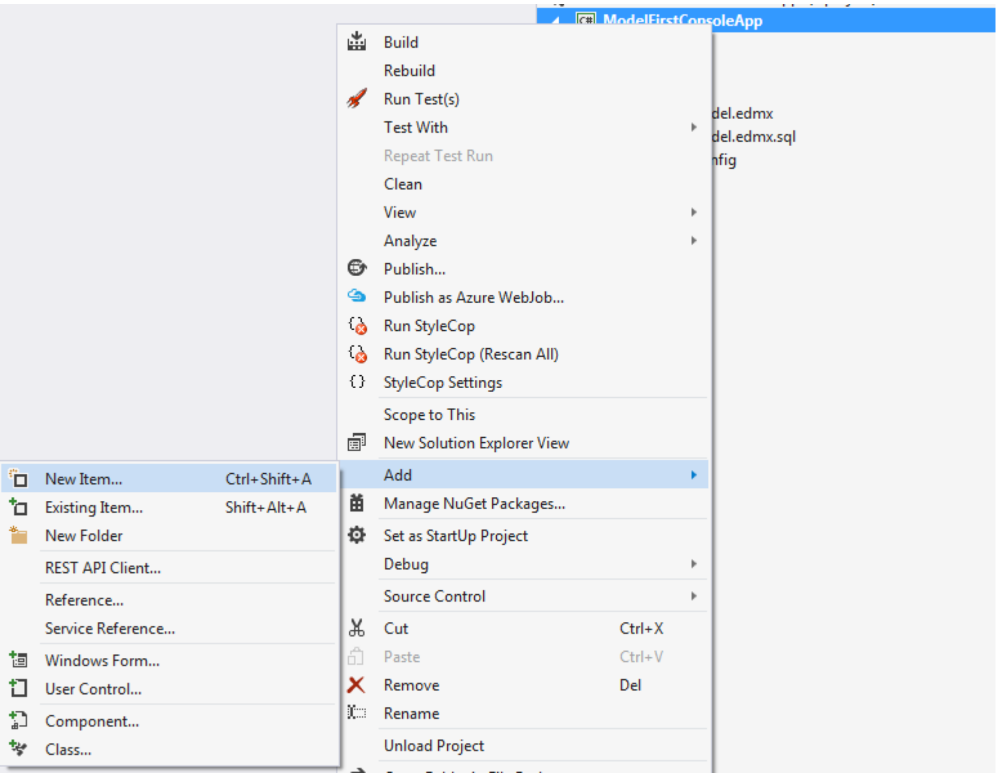
<u>Complex Type:-</u>

Complex types are non-scalar properties of entity types that enable scalar properties to be organized within entities. Like entities, complex types consist of scalar properties or other complex type properties.

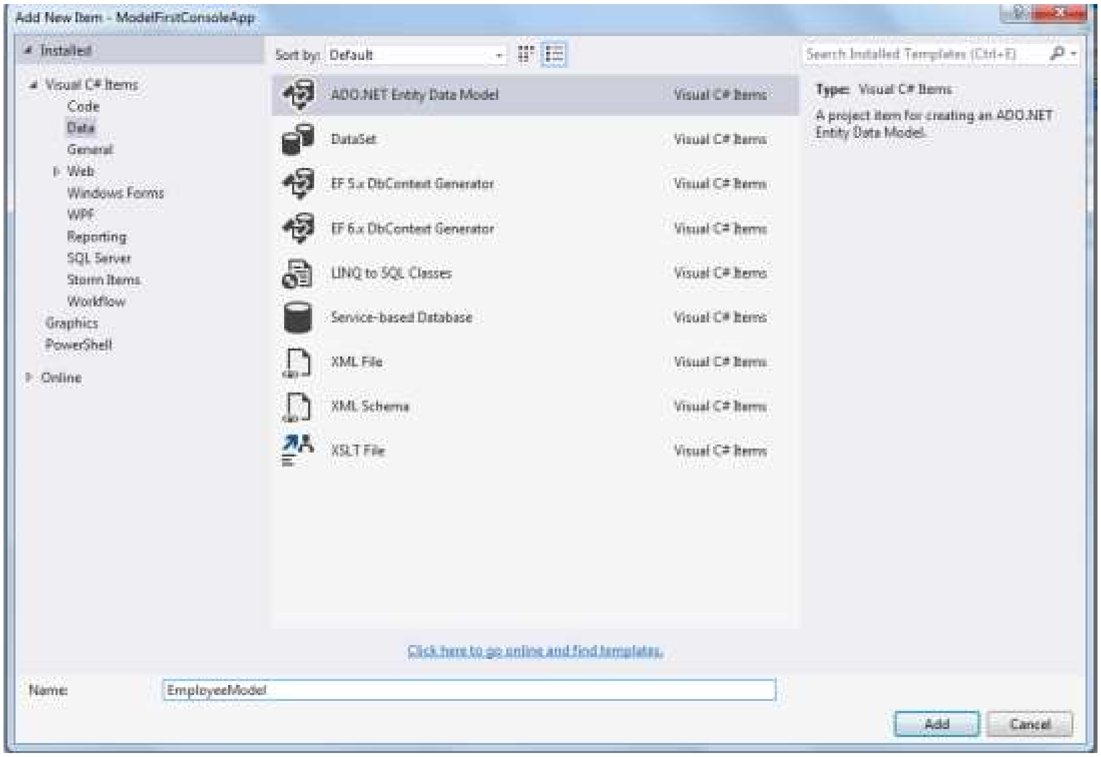
Solution:-

Creating a Complex Type using EF Designer.

In the ModelFirstConsoleApp Example, Right click on the project in solution explorer then click on Add -> New Item



In the Add New Item Dialog select ADO.Net Entity Data Model and Name it as EmployeeModel and click on Add

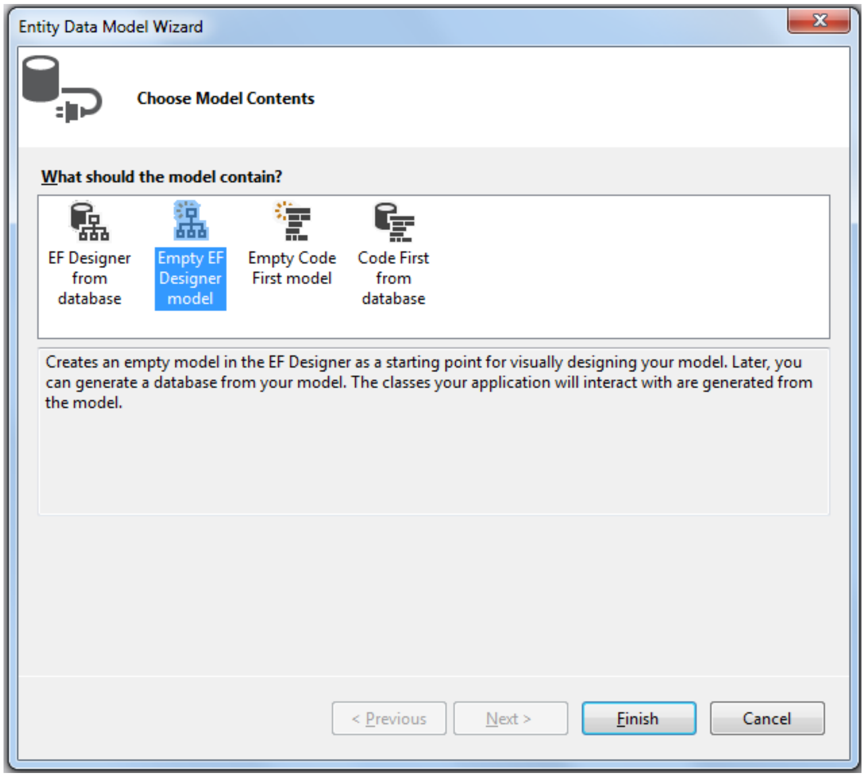

Add New Item - ModelFirstConsoleApp
Installed
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
SQL Server
Storm Items
Workflow
Graphics
PowerShell
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
ADO.NET Entity Data Model Visual C# Items
DataSet Visual C# Items
EF 5.x DbContext Generator Visual C# Items
EF 6.x DbContext Generator Visual C# Items
LINQ to SQL Classes Visual C# Items
Service-based Database Visual C# Items
XML File Visual C# Items
XML Schema Visual C# Items
XSLT File Visual C# Items
Click here to go online and find templates.
Add
Cancel

Type: Visual C# Items

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

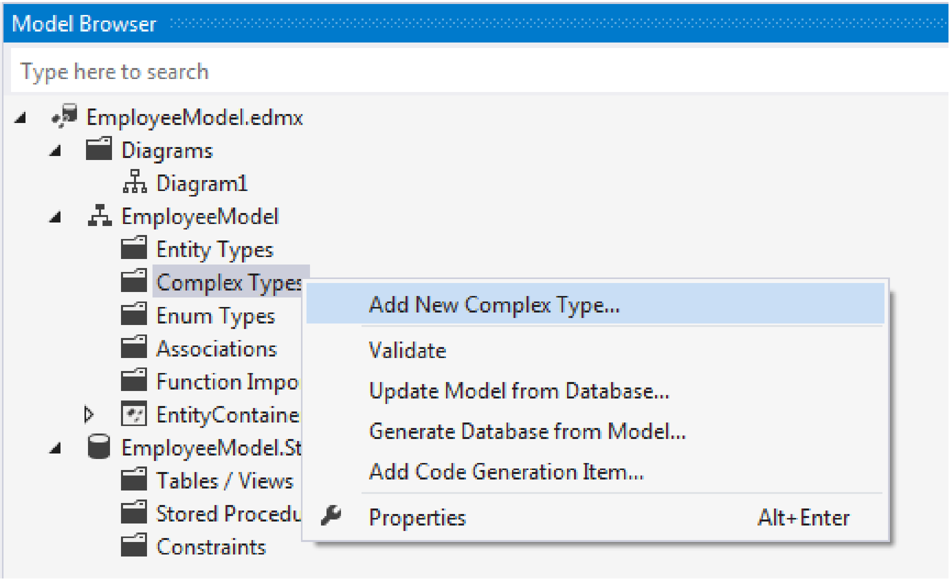
Name: EmployeeModel

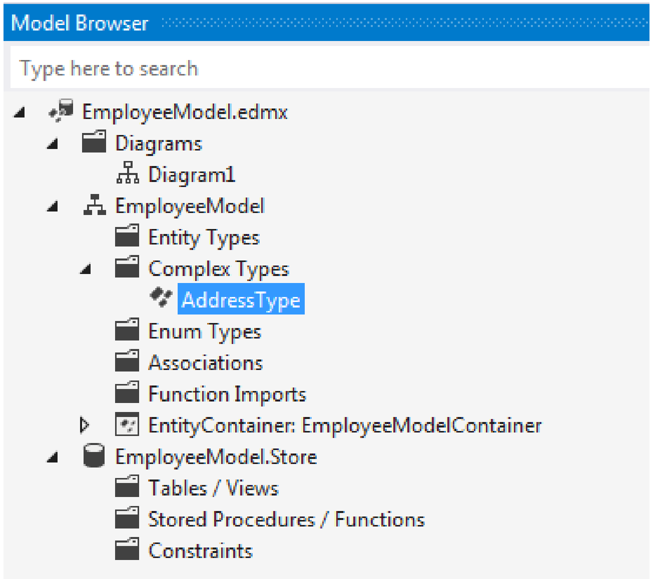
In the Entity Data Model Wizard Select Empty EF Designer Model and click on Finish.



Creates an empty model in the EF Designer as a starting point for visually designing your model. Later, you can generate a database from your model. The classes your application will interact with are generated from the model.

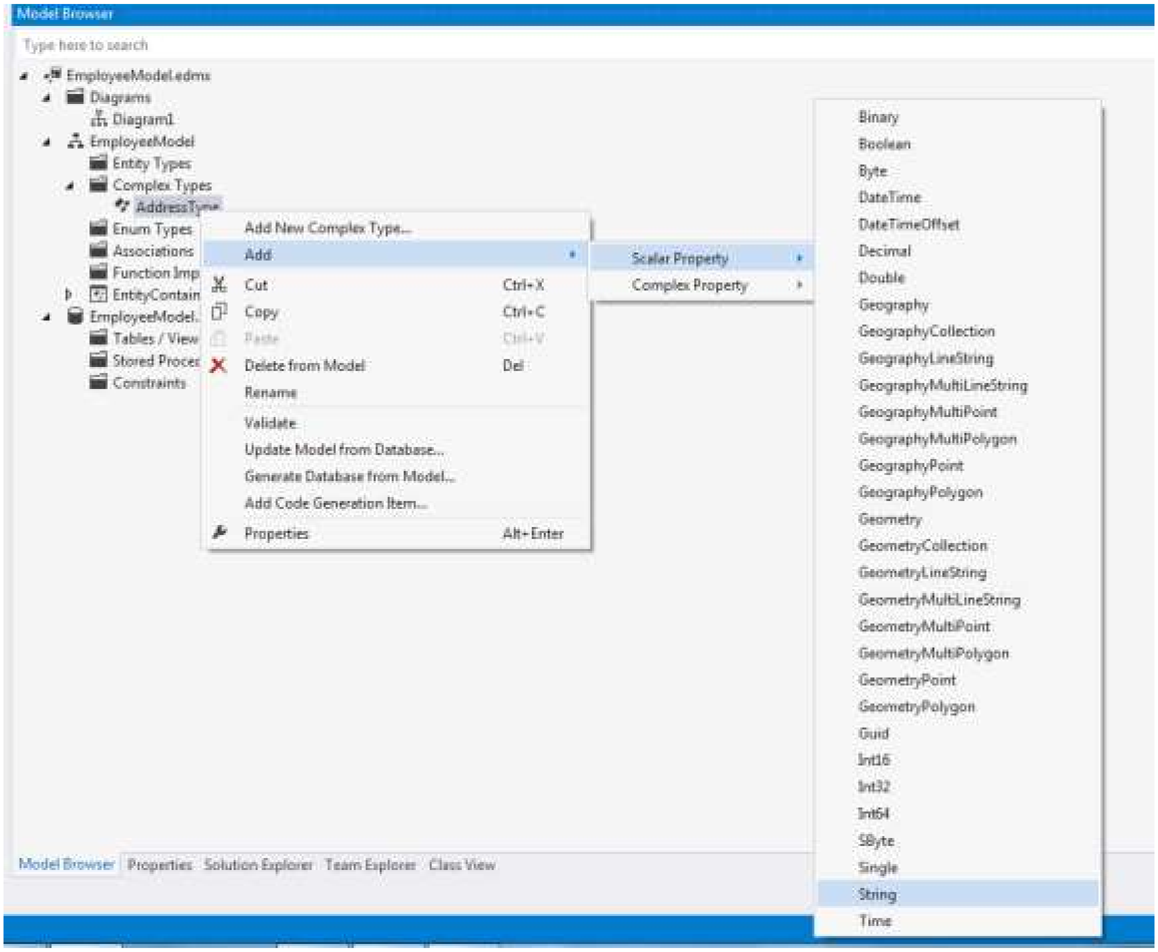
Now after adding the EmployeeModel.edmx file , In the Model Browser windows right click on the Complex Types folder of EmployeeModel and Click on Add new Complex Type and Name it as AddressType

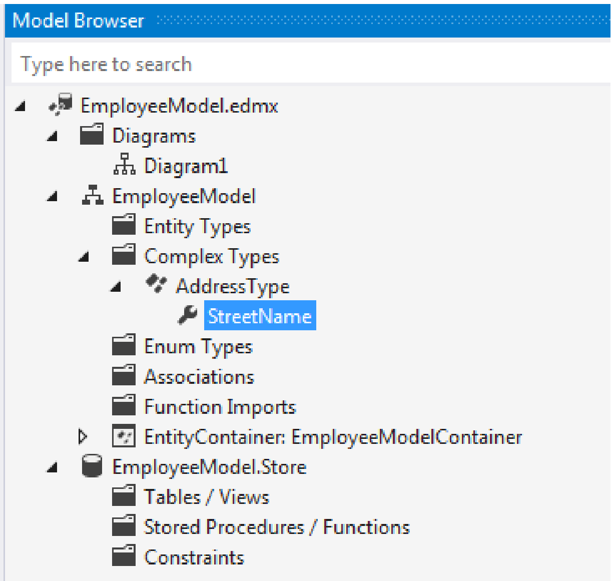




Now after creating the type Now we have to add Scalar property to it .

For Adding scalar property to the complex type right click on the complex type then Add -> Scalar Property -> Select data type of the property and then named the property as StreetName.





Model Browser

Type here to search

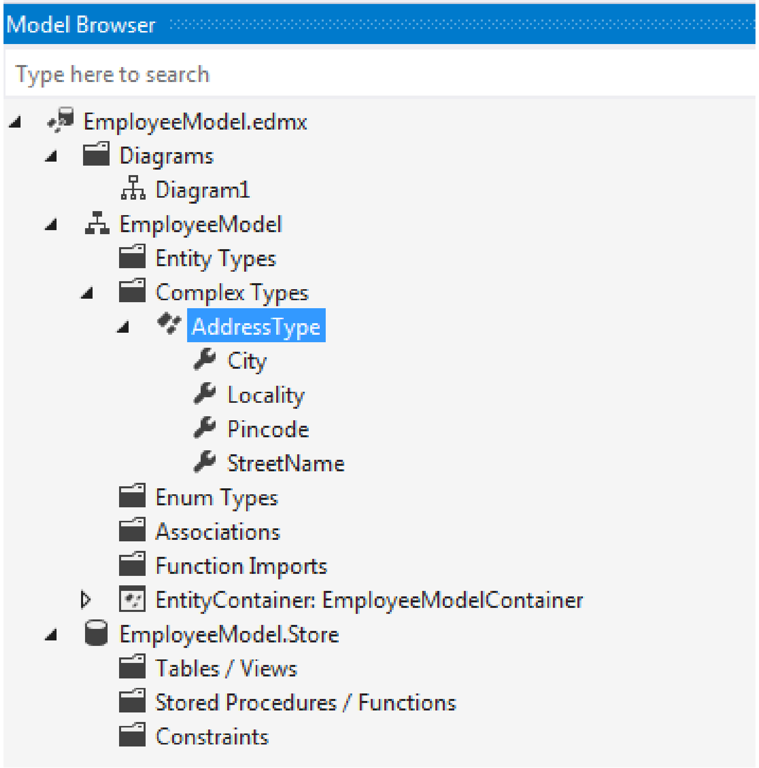
```
◢ EmployeeModel.edmx
  ◢ Diagrams
      Diagram1
  ◢ EmployeeModel
      Entity Types
    ◢ Complex Types
      ◢ AddressType
          StreetName
      Enum Types
      Associations
      Function Imports
    ▷ EntityContainer: EmployeeModelContainer
  ◢ EmployeeModel.Store
      Tables / Views
      Stored Procedures / Functions
      Constraints
```

Now Similarly Three More property to the Complex Type as follows

1) Locality

2) City

3) Pincode

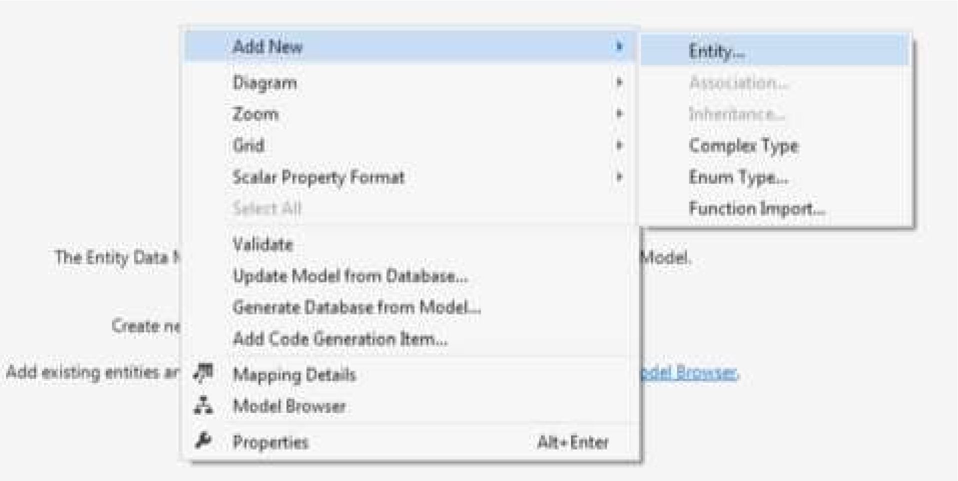


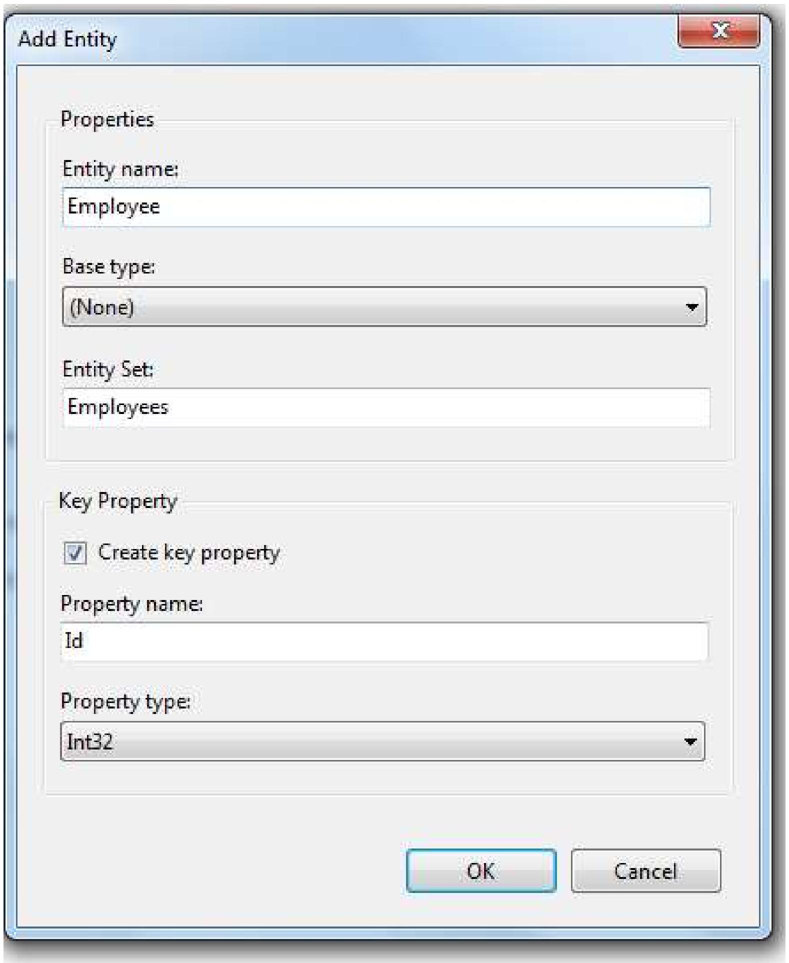
Model Browser

Type here to search

```
◢ EmployeeModel.edmx
  ◢ Diagrams
      Diagram1
  ◢ EmployeeModel
      Entity Types
    ◢ Complex Types
      ◢ AddressType
          City
          Locality
          Pincode
          StreetName
      Enum Types
      Associations
      Function Imports
    ▷ EntityContainer: EmployeeModelContainer
  ◢ EmployeeModel.Store
      Tables / Views
      Stored Procedures / Functions
      Constraints
```

Now we will add a Entity to the Designer. Right click on the designer click on Add New □ Entity and name the entity as Employee




Add Entity

Properties

Entity name: Employee

Base type: (None)

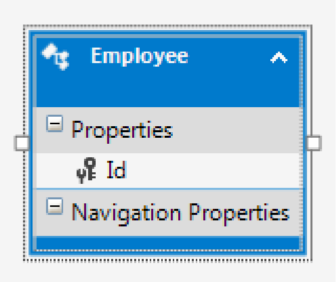
Entity Set: Employees

Key Property

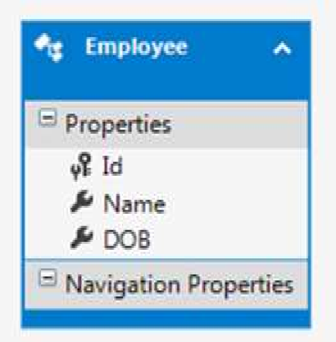
☑ Create key property

Property name: Id

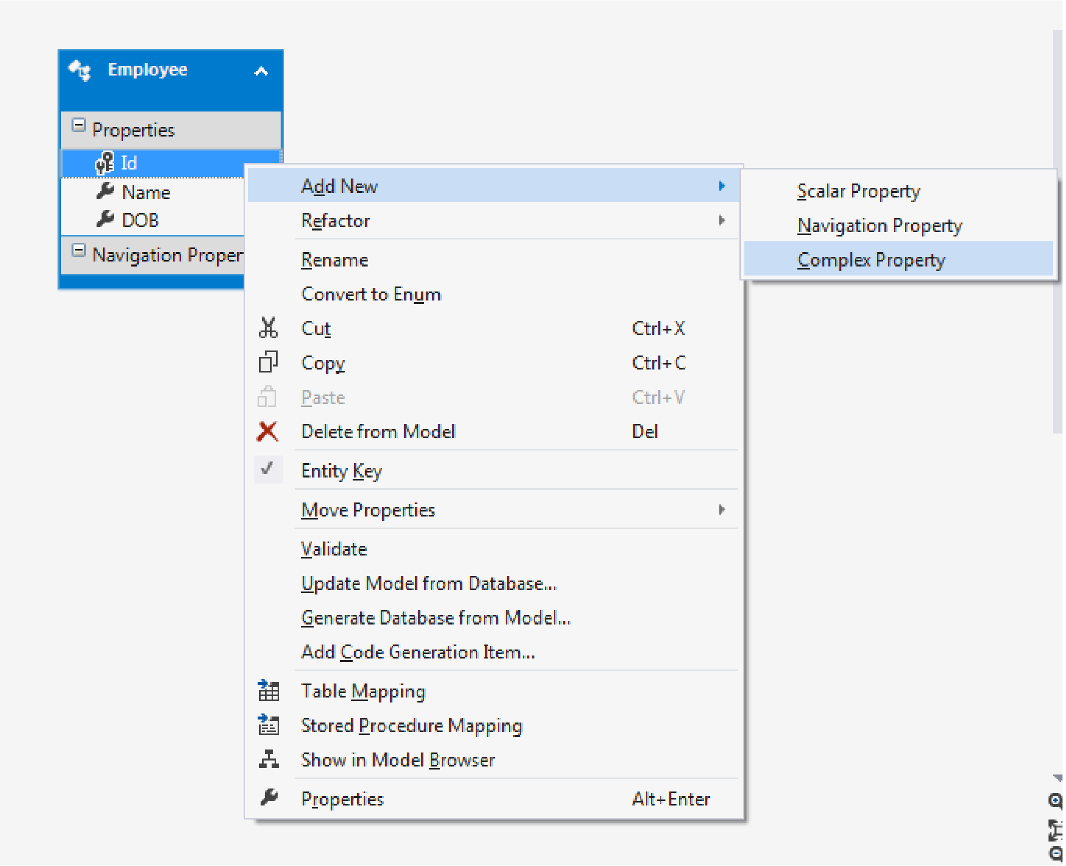
Property type: Int32

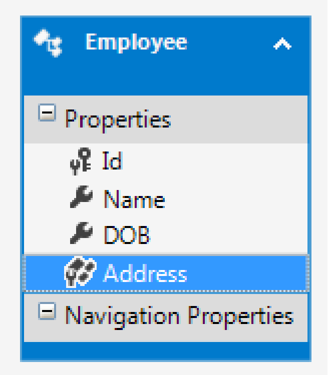


Now in the Entity add a Scalar property named as Name and DOB

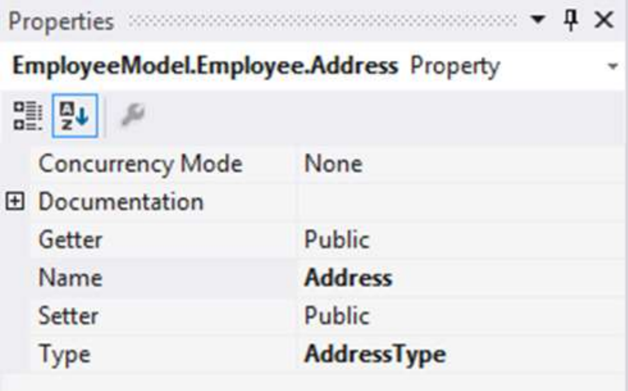


Now to Add the Complex type to the Entity right click on the entity and click on Add New -> Complex Property and name the property as Address





As we have only one Complex Type in the Model so the newly added comple property in the Entity is Model to that complex type

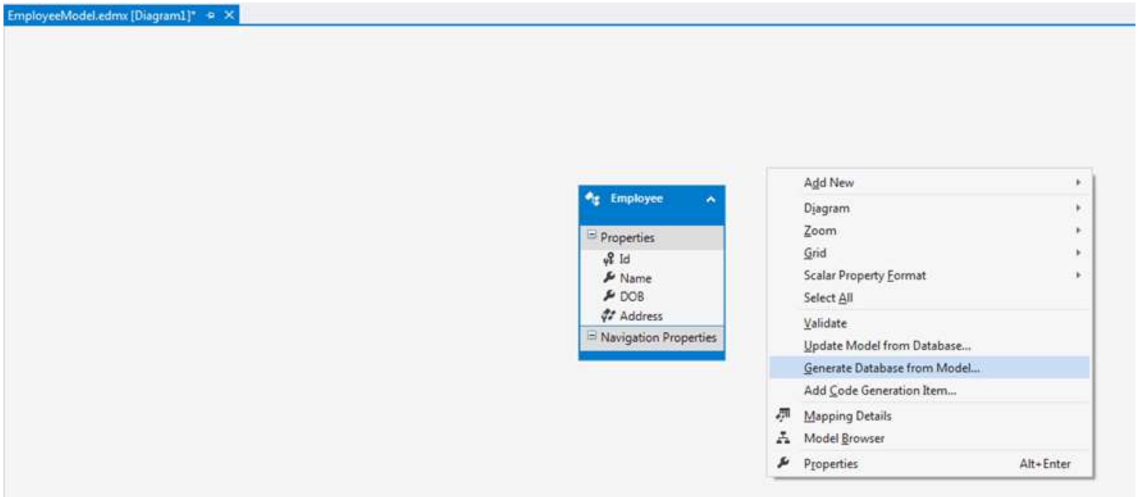

Properties

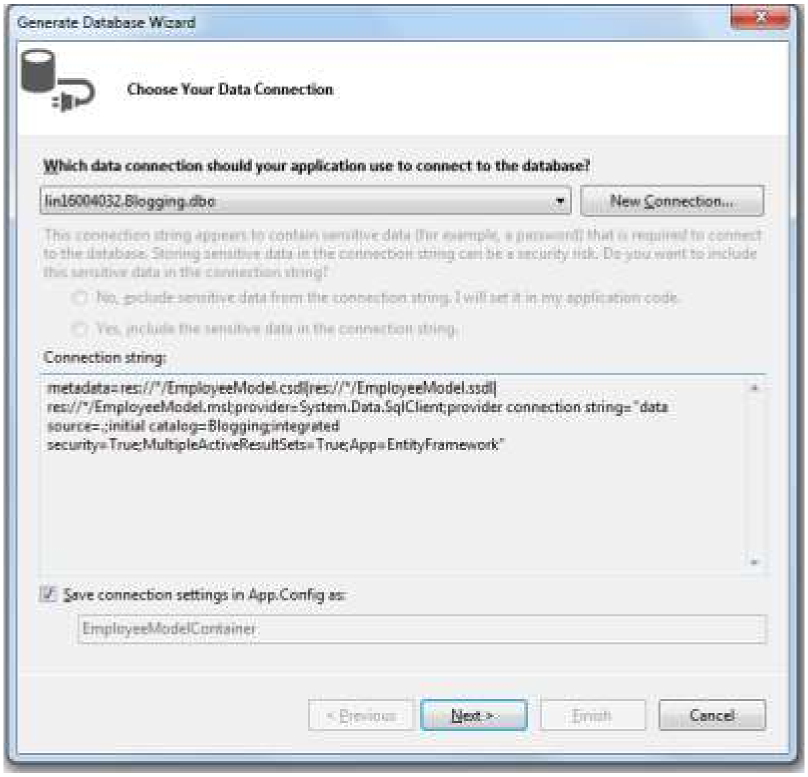
EmployeeModel.Employee.Address Property

Concurrency Mode	None
⊞ Documentation	
Getter	Public
Name	**Address**
Setter	Public
Type	**AddressType**

Now we have to generate database from the model for that Right click on the designer and click on the Generate Database from Model.

In this we will use the connection that we have created for Blogging database.





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

lin16004032.Blogging.dbo

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.

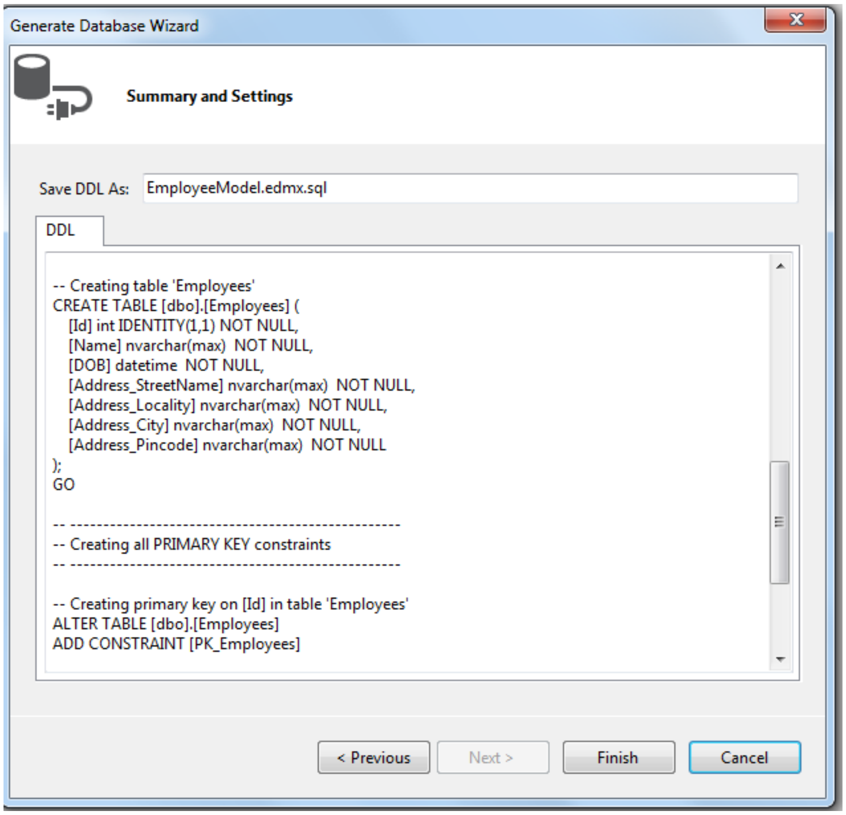
○ Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/EmployeeModel.csdl|res://*/EmployeeModel.ssdl|
res://*/EmployeeModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=.;initial catalog=Blogging;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in App.Config as:

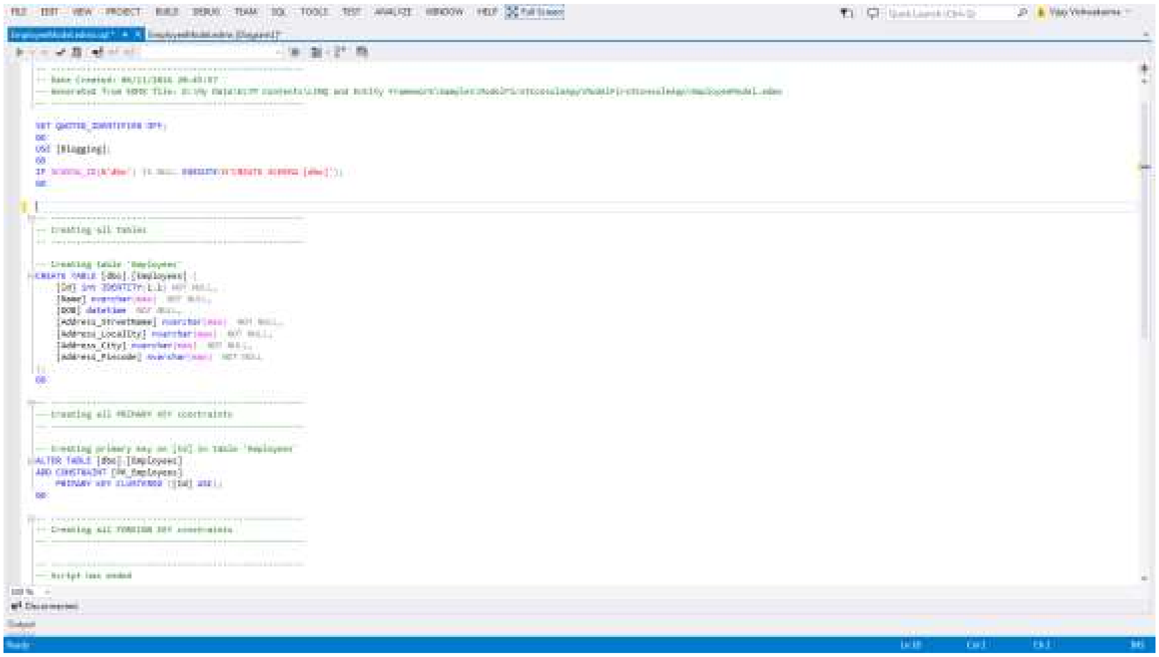
EmployeeModelContainer



Save DDL As: EmployeeModel.edmx.sql

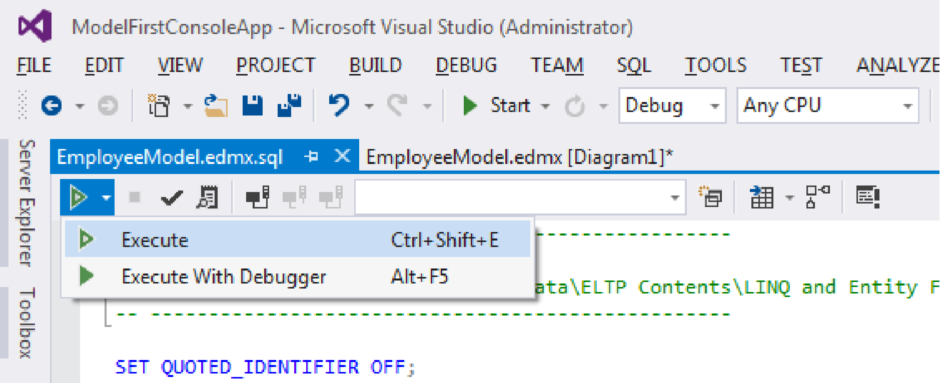
```
-- Creating table 'Employees'
CREATE TABLE [dbo].[Employees] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(max)  NOT NULL,
    [DOB] datetime  NOT NULL,
    [Address_StreetName] nvarchar(max)  NOT NULL,
    [Address_Locality] nvarchar(max)  NOT NULL,
    [Address_City] nvarchar(max)  NOT NULL,
    [Address_Pincode] nvarchar(max)  NOT NULL
);
GO

-- --------------------------------------------------
-- Creating all PRIMARY KEY constraints
-- --------------------------------------------------

-- Creating primary key on [Id] in table 'Employees'
ALTER TABLE [dbo].[Employees]
ADD CONSTRAINT [PK_Employees]
```

Click on Finish this will create a Sql file named as EmployeeModel.edmx.sql which we will execute against database for creating database and tables .



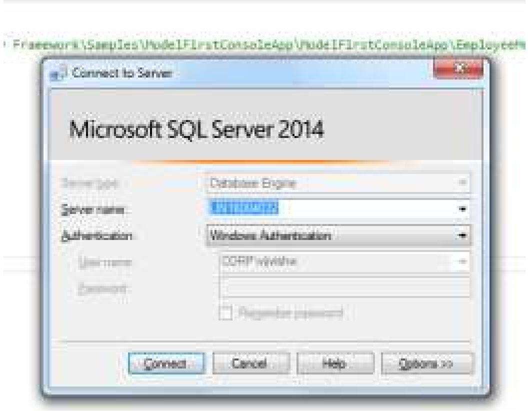
```
-- --------------------------------------------------
-- Date Created: 08/11/2016 20:45:07
-- Generated from EDMX file: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\ModelFirstConsoleApp\ModelFirstConsoleApp\EmployeeModel.edmx
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
GO
USE [Blogging];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

|
-- --------------------------------------------------
-- Creating all tables
-- --------------------------------------------------

-- Creating table 'Employees'
CREATE TABLE [dbo].[Employees] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(max)  NOT NULL,
    [DOB] datetime  NOT NULL,
    [Address_StreetName] nvarchar(max)  NOT NULL,
    [Address_Locality] nvarchar(max)  NOT NULL,
    [Address_City] nvarchar(max)  NOT NULL,
    [Address_Pincode] nvarchar(max)  NOT NULL
);
GO

-- --------------------------------------------------
-- Creating all PRIMARY KEY constraints
-- --------------------------------------------------

-- Creating primary key on [Id] in table 'Employees'
ALTER TABLE [dbo].[Employees]
ADD CONSTRAINT [PK_Employees]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- --------------------------------------------------
-- Creating all FOREIGN KEY constraints
-- --------------------------------------------------

-- --------------------------------------------------
-- Script has ended
```

Now click on the Execute button to execute the sql query against the database.



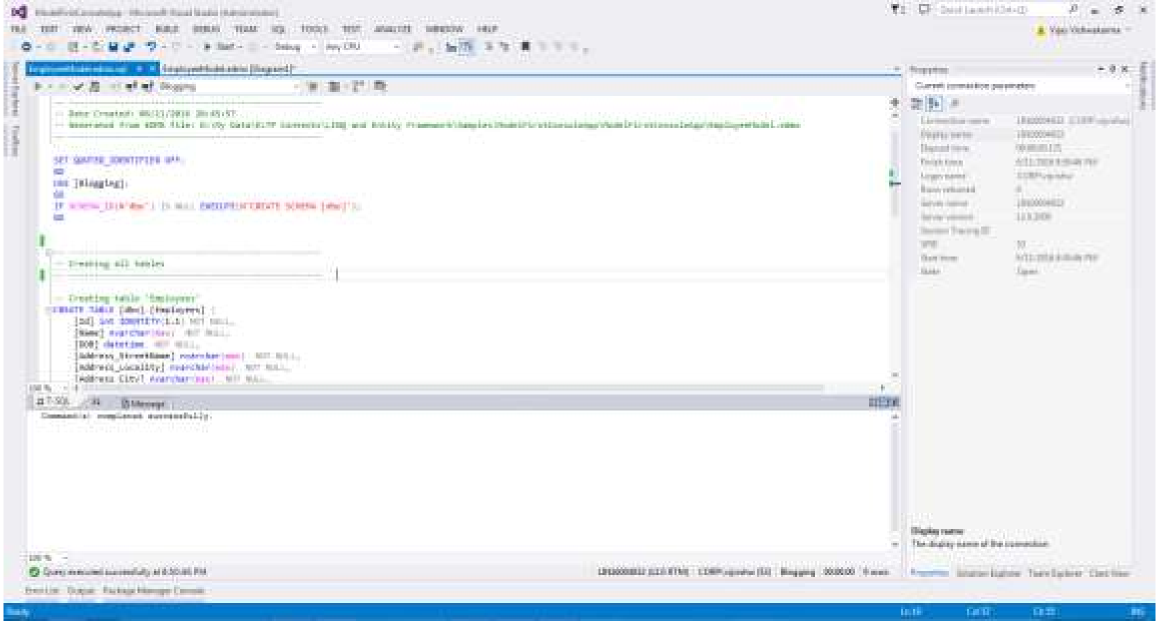
```
-- --------------------------------------------------
ata\ELTP Contents\LINQ and Entity F
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
```

Connect to database



```
Framework\Samples\ModelFirstConsoleApp\ModelFirstConsoleApp\EmployeeM
```

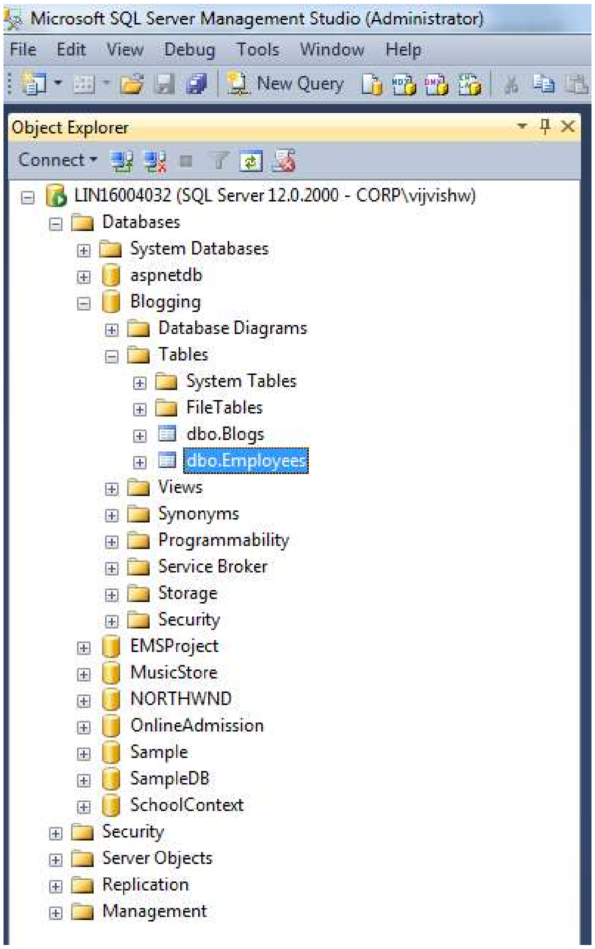
Server type: Database Engine
Server name: [illegible]
Authentication: Windows Authentication
User name: CORP\[illegible]
Password:
□ Remember password



```
-- Date Created: 05/11/2016 20:45:57
-- Generated from EDMX file: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\ModelFirstConsoleApp\ModelFirstConsoleApp\EmployeeModel.edmx

SET QUOTED_IDENTIFIER OFF;
GO
USE [Blogging];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

-- Creating all tables

-- Creating table 'Employees'
CREATE TABLE [dbo].[Employees] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(max) NOT NULL,
    [DOB] datetime NOT NULL,
    [Address_StreetName] nvarchar(max) NOT NULL,
    [Address_Locality] nvarchar(max) NOT NULL,
    [Address_City] nvarchar(max) NOT NULL,
```

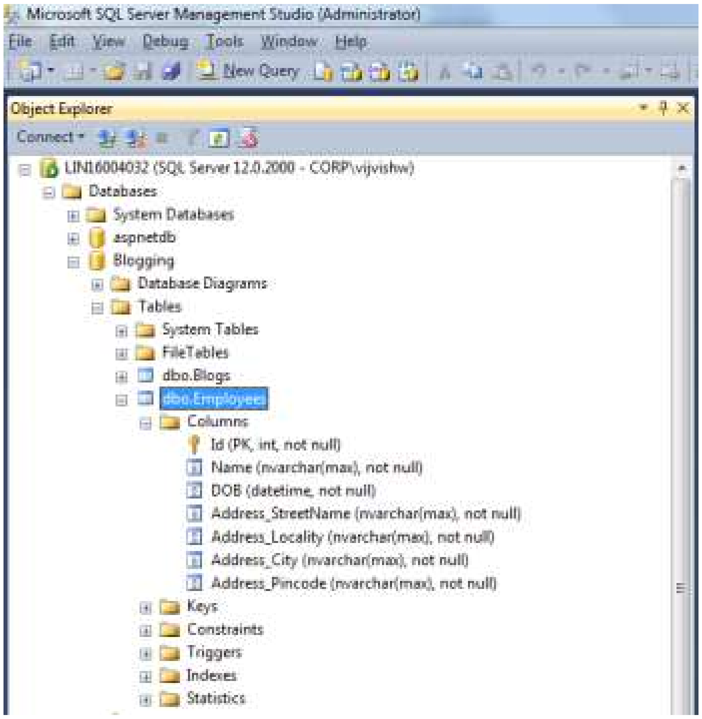
```
Command(s) completed successfully.
```

Now to view the table open Sql Server Management Studio and in the Object Explorer expand blogging database and then expand table tab to view the newly created table



Microsoft SQL Server Management Studio (Administrator)
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect

```
LIN16004032 (SQL Server 12.0.2000 - CORP\vijvishw)
  Databases
    System Databases
    aspnetdb
    Blogging
      Database Diagrams
      Tables
        System Tables
        FileTables
        dbo.Blogs
        dbo.Employees
      Views
      Synonyms
      Programmability
      Service Broker
      Storage
      Security
    EMSProject
    MusicStore
    NORTHWND
    OnlineAdmission
    Sample
    SampleDB
    SchoolContext
  Security
  Server Objects
  Replication
  Management
```


Microsoft SQL Server Management Studio (Administrator)
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect

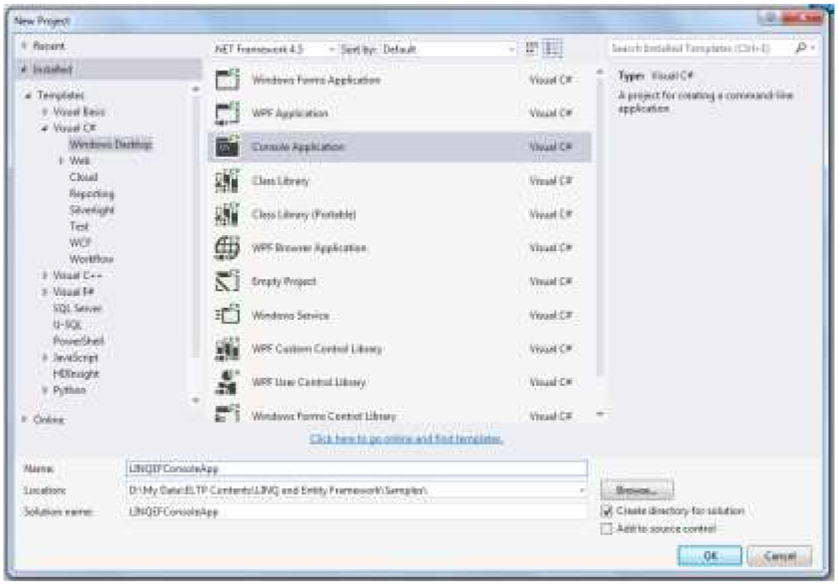
```
LIN16004032 (SQL Server 12.0.2000 - CORP\vijvishw)
  Databases
    System Databases
    aspnetdb
    Blogging
      Database Diagrams
      Tables
        System Tables
        FileTables
        dbo.Blogs
        dbo.Employees
          Columns
            Id (PK, int, not null)
            Name (nvarchar(max), not null)
            DOB (datetime, not null)
            Address_StreetName (nvarchar(max), not null)
            Address_Locality (nvarchar(max), not null)
            Address_City (nvarchar(max), not null)
            Address_Pincode (nvarchar(max), not null)
          Keys
          Constraints
          Triggers
          Indexes
          Statistics
```

Lab 5. Basic Query Operations using LINQ to Entities

Goals	Understand the process of performing LINQ queries on Entity Models Learn to use of LINQ to Entities Learn to Manipulate Data
Time	60 minutes

Solution:-

Open Visual studio and create a new console application named as LINQEFConsoleApp and the Entity Framework Library using the Nuget Package Manager to the application.


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows Desktop
Web
Cloud
Reporting
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
U-SQL
PowerShell
JavaScript
HDInsight
Python
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
Windows Service Visual C#
WPF Custom Control Library Visual C#
WPF User Control Library Visual C#
Windows Forms Control Library Visual C#
Click here to go online and find templates.
Browse...
OK
Cancel

Type: Visual C#

A project for creating a command-line application

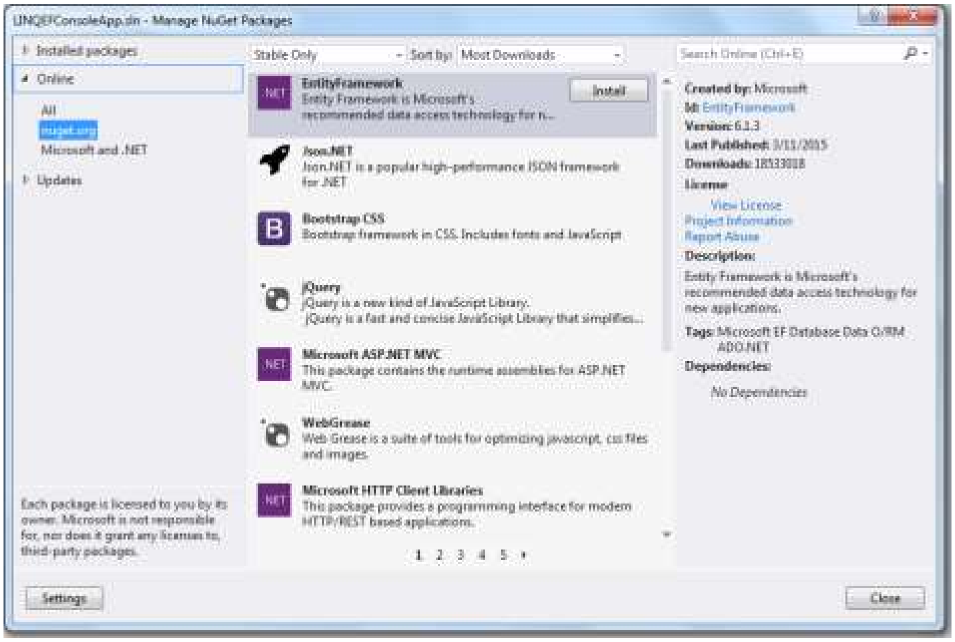
Name: LINQEFConsoleApp

Location: D:\My Data\ELTP Contents\LINQ and Entity Framework\Samples\

Solution name: LINQEFConsoleApp

☑ Create directory for solution

☐ Add to source control



EntityFramework
Entity Framework is Microsoft's recommended data access technology for n...

Json.NET
Json.NET is a popular high-performance JSON framework for .NET

Bootstrap CSS
Bootstrap framework in CSS. Includes fonts and JavaScript

jQuery
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies...

Microsoft ASP.NET MVC
This package contains the runtime assemblies for ASP.NET MVC.

WebGrease
Web Grease is a suite of tools for optimizing javascript, css files and images.

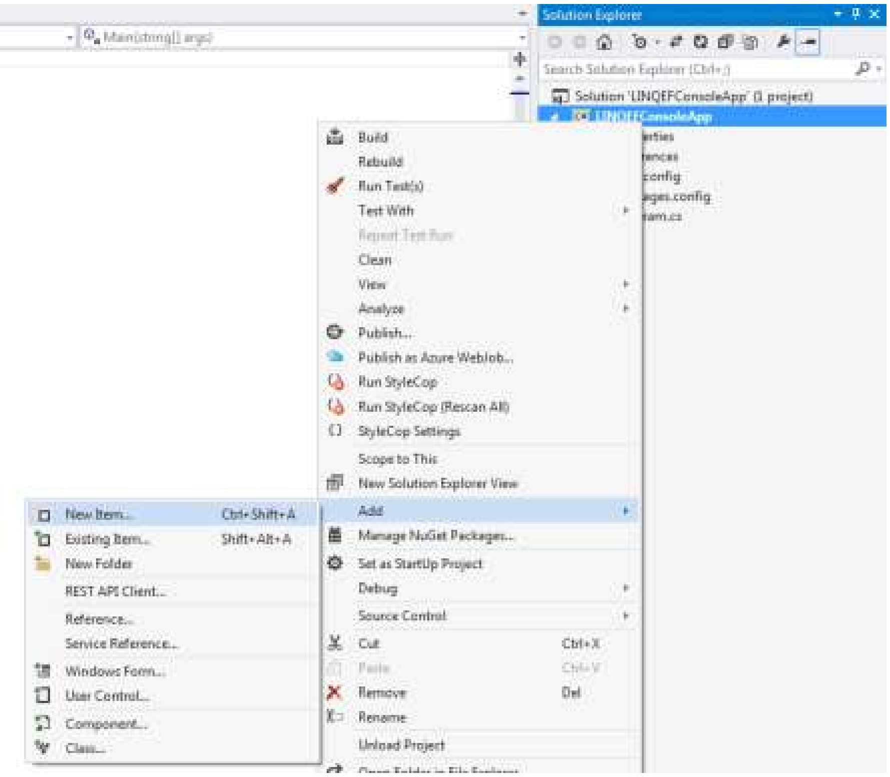
Microsoft HTTP Client Libraries
This package provides a programming interface for modern HTTP/REST based applications.

Created by: Microsoft
Id: EntityFramework
Version: 6.1.3
Last Published: 3/11/2015
Downloads: 18533018
License
View License
Project Information
Report Abuse
Description:
Entity Framework is Microsoft's recommended data access technology for new applications.
Tags: Microsoft EF Database Data O/RM ADO.NET
Dependencies:
No Dependencies

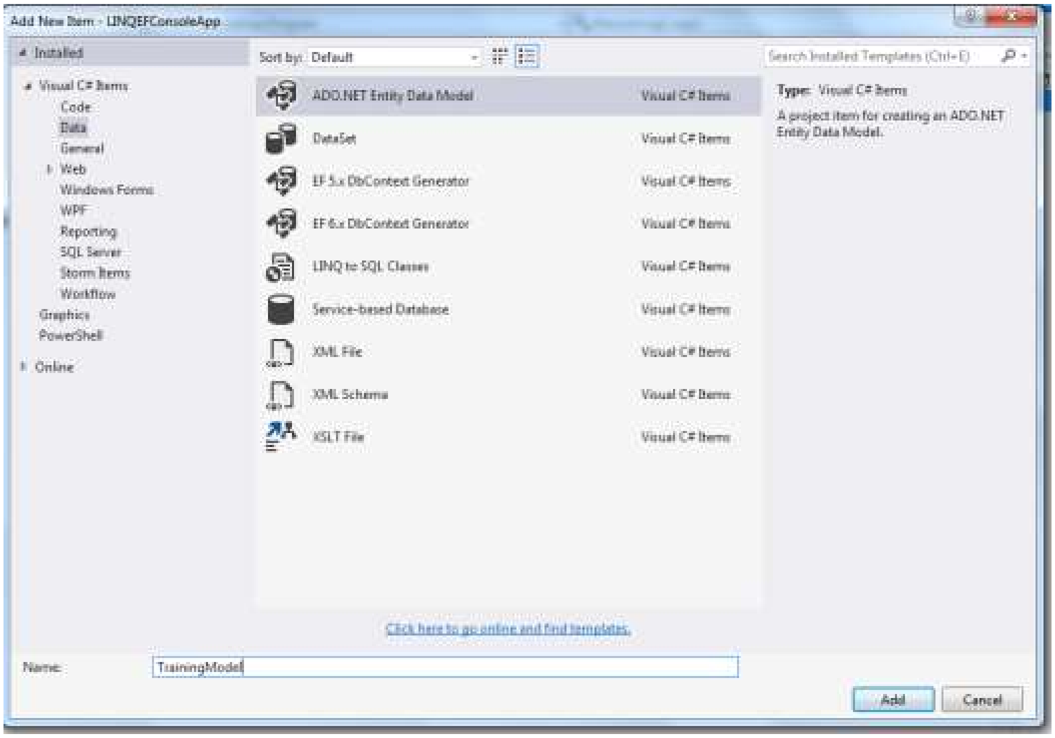
Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Now we have to create a Entity Data Model from an existing database.

Right click on the project in the solution explorer and select Add -> New Item



In the new Item Dialog Box select ADO.Net Entity Data Model and name it as Training Model

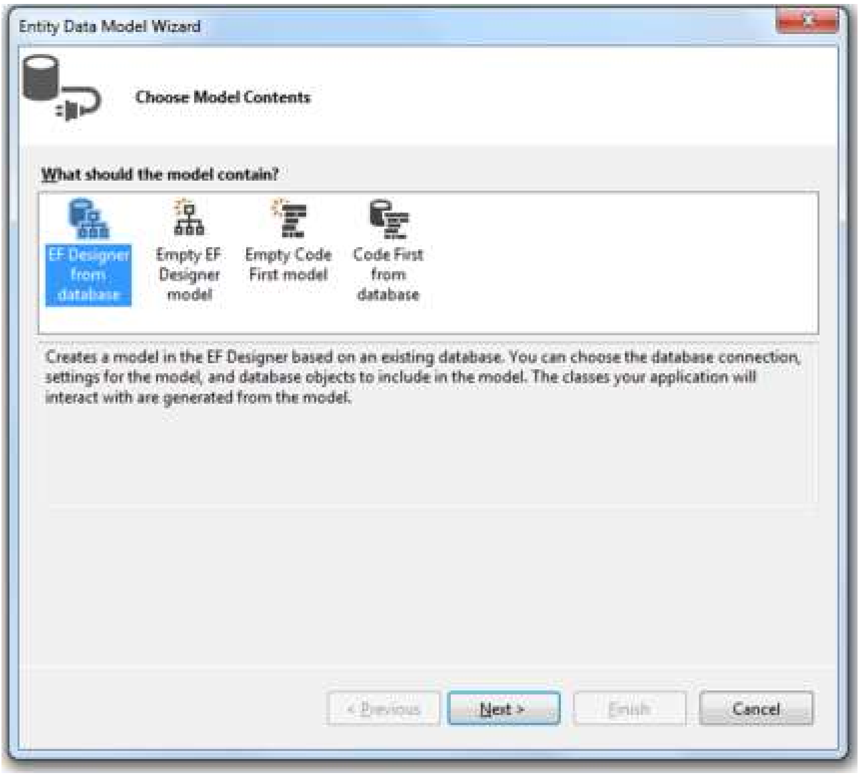

Add New Item - LINQEFConsoleApp
Installed
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
Reporting
SQL Server
Storm Items
Workflow
Graphics
PowerShell
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
ADO.NET Entity Data Model Visual C# Items
DataSet Visual C# Items
EF 5.x DbContext Generator Visual C# Items
EF 6.x DbContext Generator Visual C# Items
LINQ to SQL Classes Visual C# Items
Service-based Database Visual C# Items
XML File Visual C# Items
XML Schema Visual C# Items
XSLT File Visual C# Items
Click here to go online and find templates.
Add
Cancel

Type: Visual C# Items

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

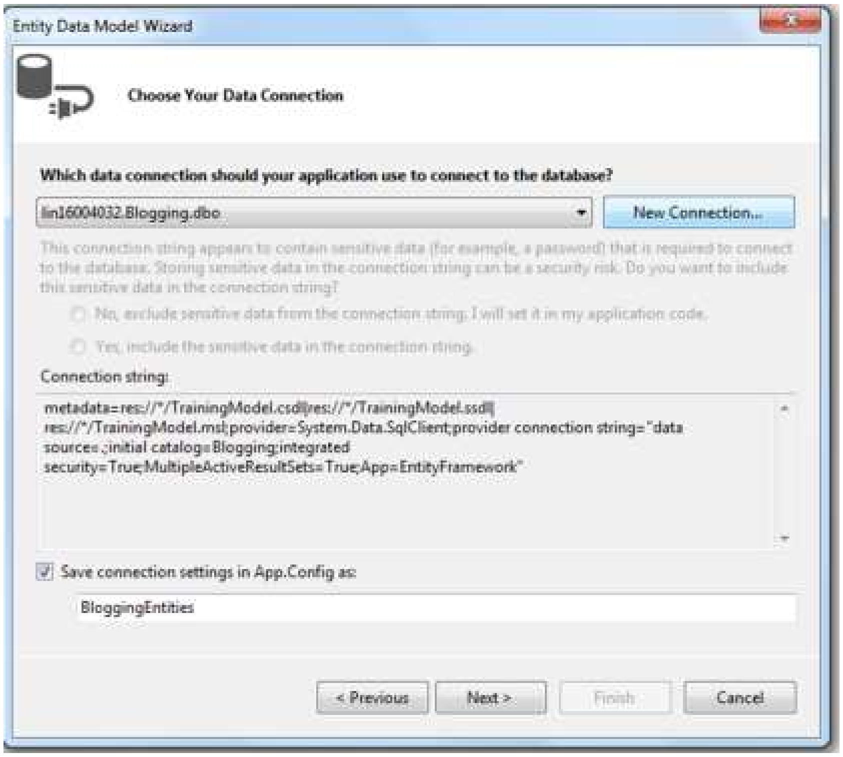
Name: TrainingModel

In the Entity Data Model Wizard select EF Designer from database and click on Next and configure the datasource.



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.

Now Click on New Connection in the Choose Your Data source option



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

lin16004032.Blogging.dbo

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.

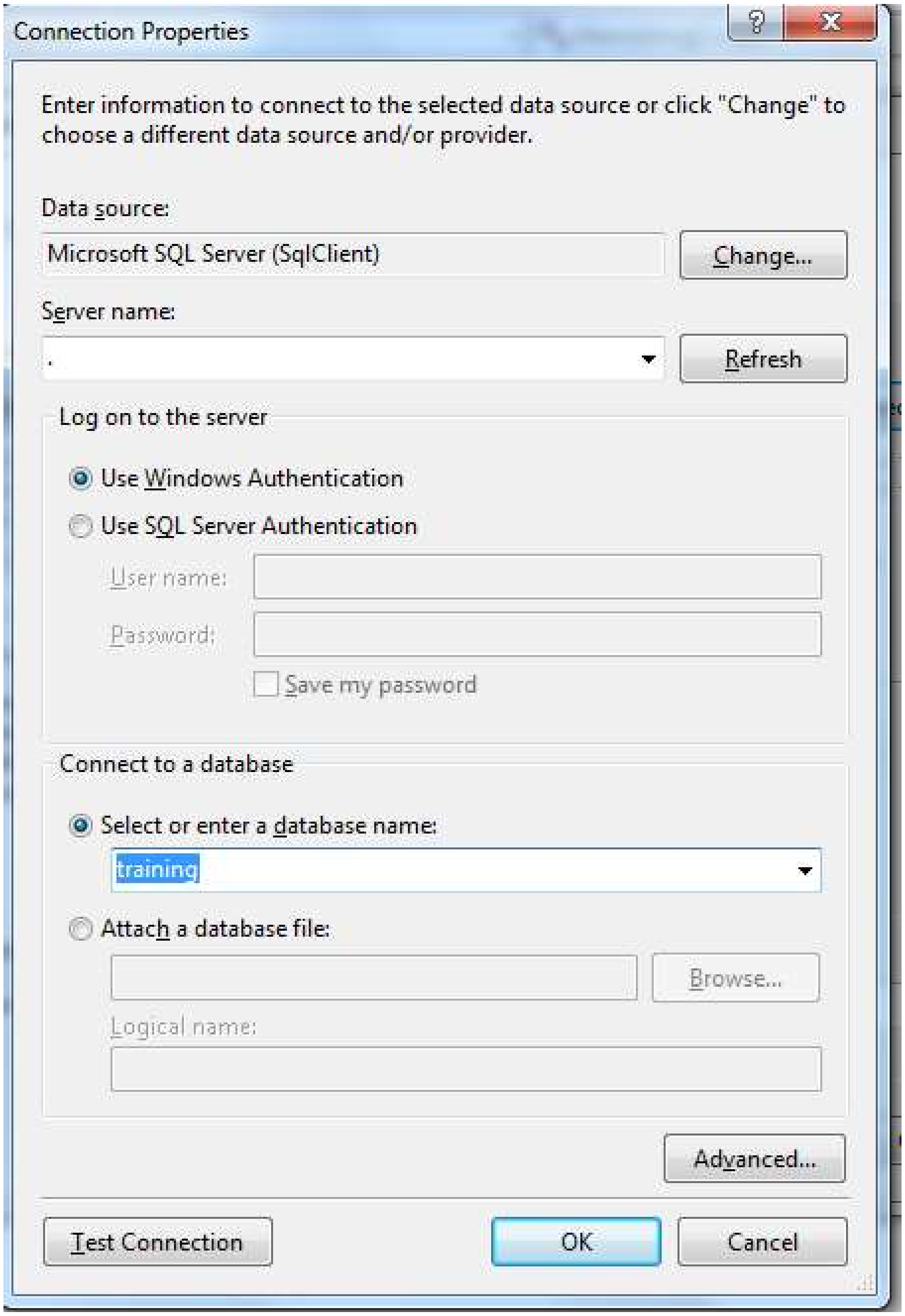
Connection string:

```
metadata=res://*/TrainingModel.csdl|res://*/TrainingModel.ssdl|
res://*/TrainingModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=.;initial catalog=Blogging;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in App.Config as:

BloggingEntities

In the connection properties windows Provide the sql server name , authentication type and database name click on Test Connection to test the connection .



Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: .

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

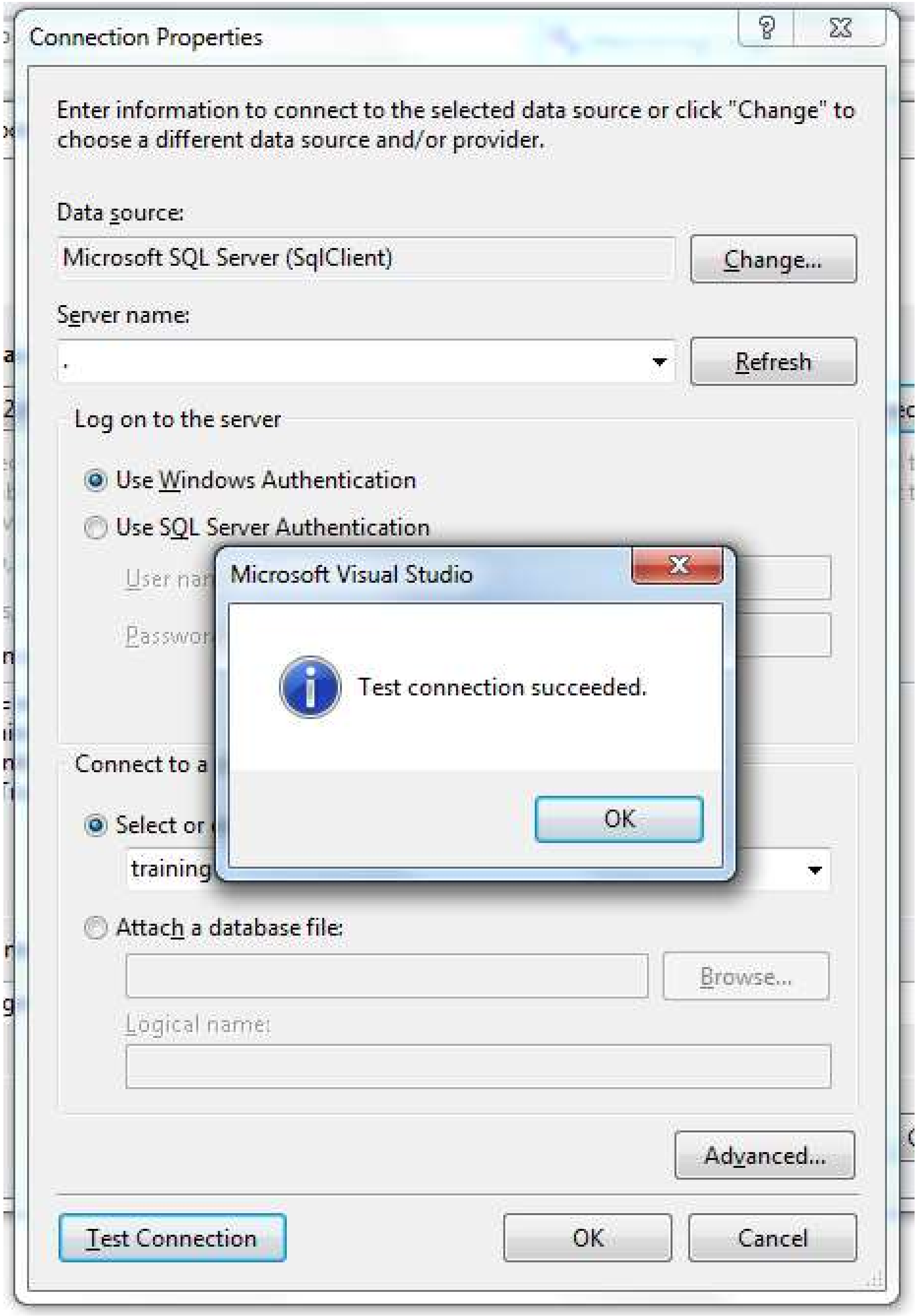
□ Save my password

Connect to a database

● Select or enter a database name: training

○ Attach a database file:

Logical name:



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: .

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

Connect to a

● Select or

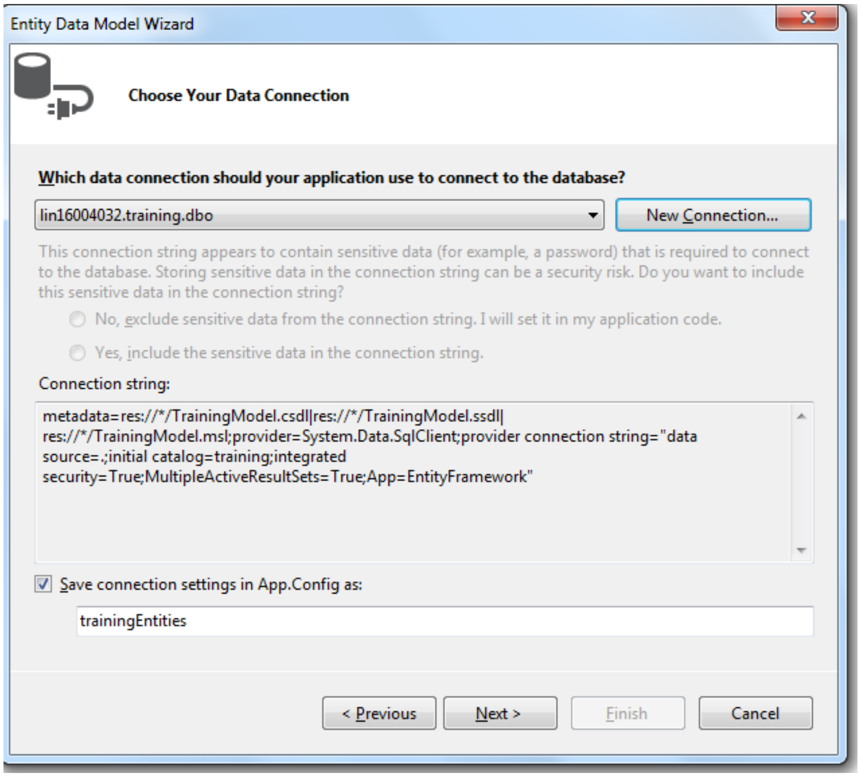
training

○ Attach a database file:

Microsoft Visual Studio

Test connection succeeded.

Click on OK



Choose Your Data Connection

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

lin16004032.training.dbo

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.

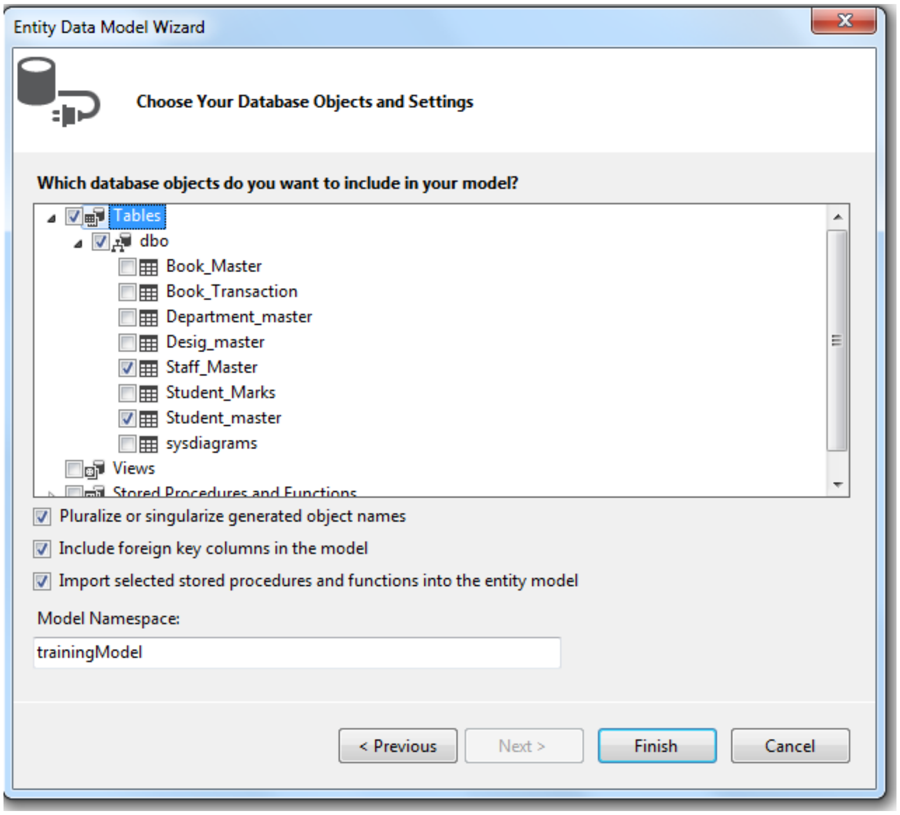
Connection string:

```
metadata=res://*/TrainingModel.csdl|res://*/TrainingModel.ssdl|
res://*/TrainingModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=.;initial catalog=training;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in App.Config as:

trainingEntities

Now click on next to select the database objects which will be part of Entity Data Model.



Choose Your Database Objects and Settings

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

- ☑ Tables
  - ☑ dbo
    - □ Book_Master
    - □ Book_Transaction
    - □ Department_master
    - □ Desig_master
    - ☑ Staff_Master
    - □ Student_Marks
    - ☑ Student_master
    - □ sysdiagrams
- □ Views
- □ Stored Procedures and Functions

☑ Pluralize or singularize generated object names

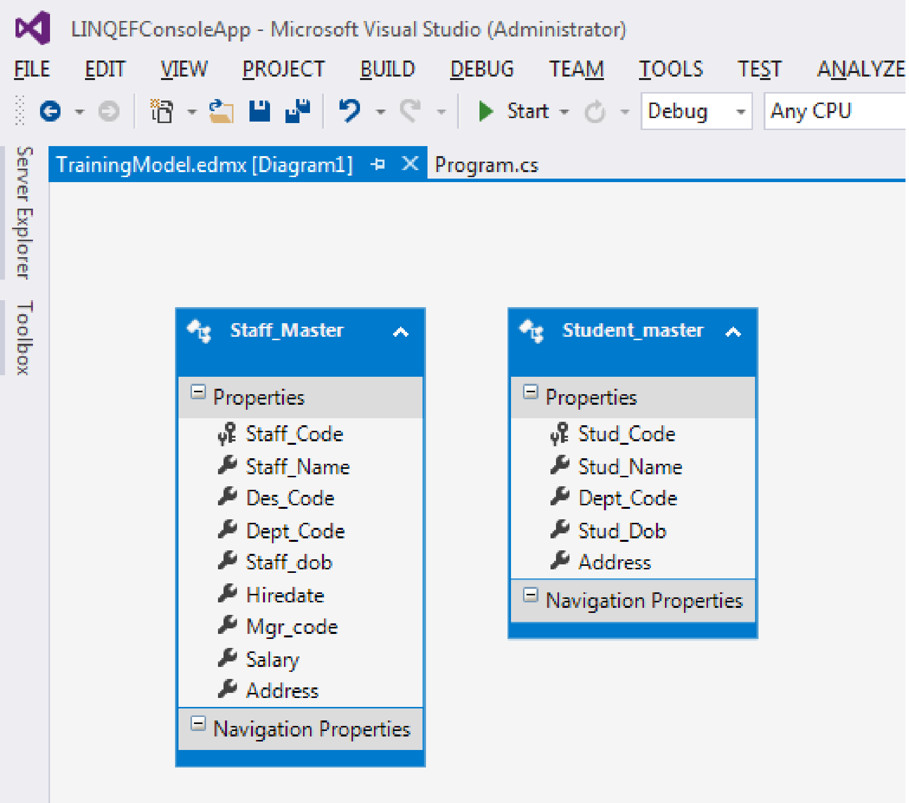
☑ Include foreign key columns in the model

☑ Import selected stored procedures and functions into the entity model

Model Namespace:

trainingModel

Tick the Staff_Master and Student_Master Table and click on Finish this will add the entities to the model.



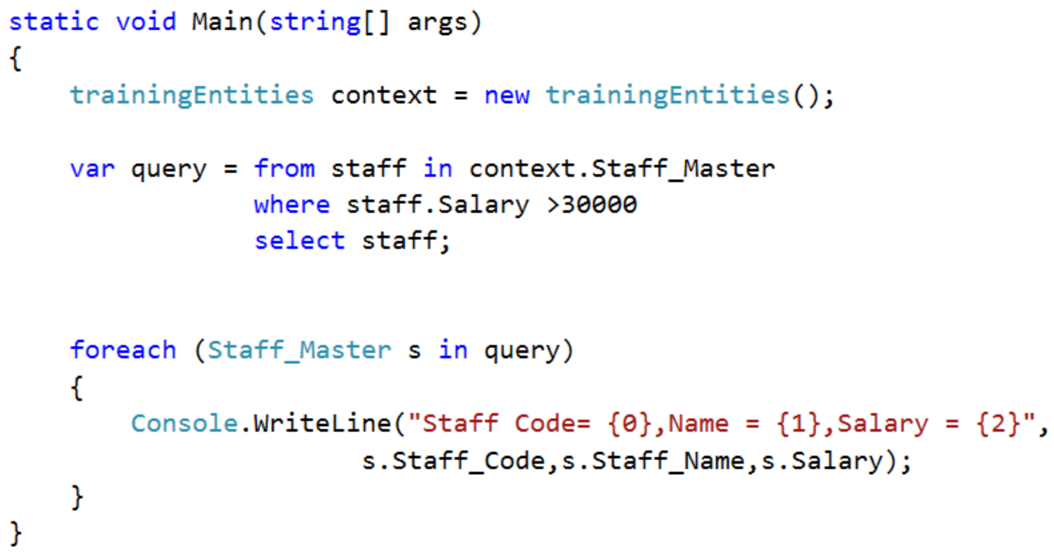
Now we have to write LINQ query against the model for reading the data.

Write Linq queries for the following.

1) To display staff details write the following query

```
static void Main(string[] args)
{
    trainingEntities context = new trainingEntities();

    var query = from staff in context.Staff_Master
                select staff;

    //Displaying details from Staff_Master Entity
    foreach (Staff_Master s in query)
    {
        Console.WriteLine("Staff Code= {0},Name = {1},HireDate = {2}",
                       s.Staff_Code,s.Staff_Name,s.Hiredate);
    }
}
```

2) To display list of employee whose salary is more than 30000



```
static void Main(string[] args)
{
    trainingEntities context = new trainingEntities();

    var query = from staff in context.Staff_Master
                where staff.Salary >30000
                select staff;


    foreach (Staff_Master s in query)
    {
        Console.WriteLine("Staff Code= {0},Name = {1},Salary = {2}",
                          s.Staff_Code,s.Staff_Name,s.Salary);
    }
}
```

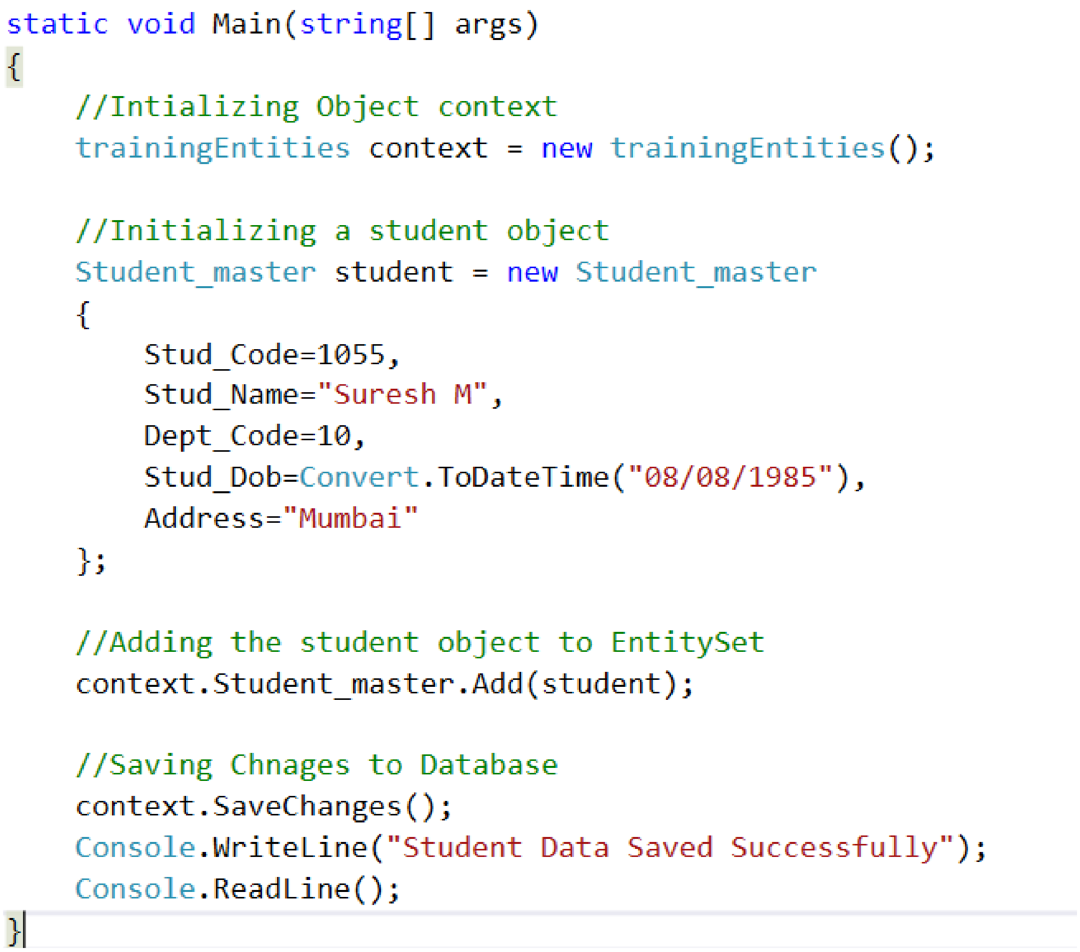
Perform the following query by yourself

3) Display the list of student where city is not null
4) Display the list of student which includes Student name, department and date of birth
5) Display count of total student belonging to Bangalore
6) Display list of employees whose salary is more than the average salary of the employee.

Data Manipulation:-

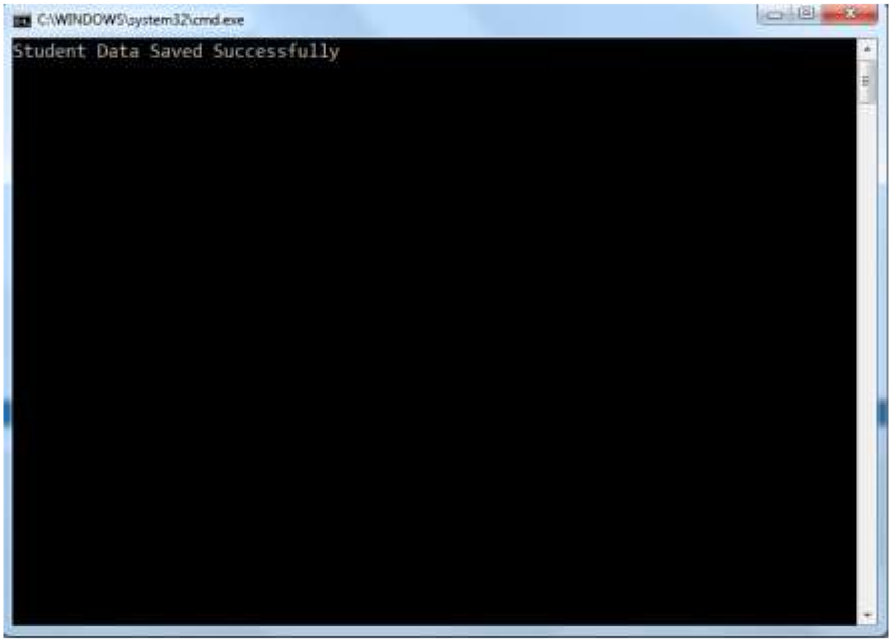
Now we will CRUD operation on the model that we have create . we will use Student_Master Entity.

To ADD a record to the student master entity write the following code

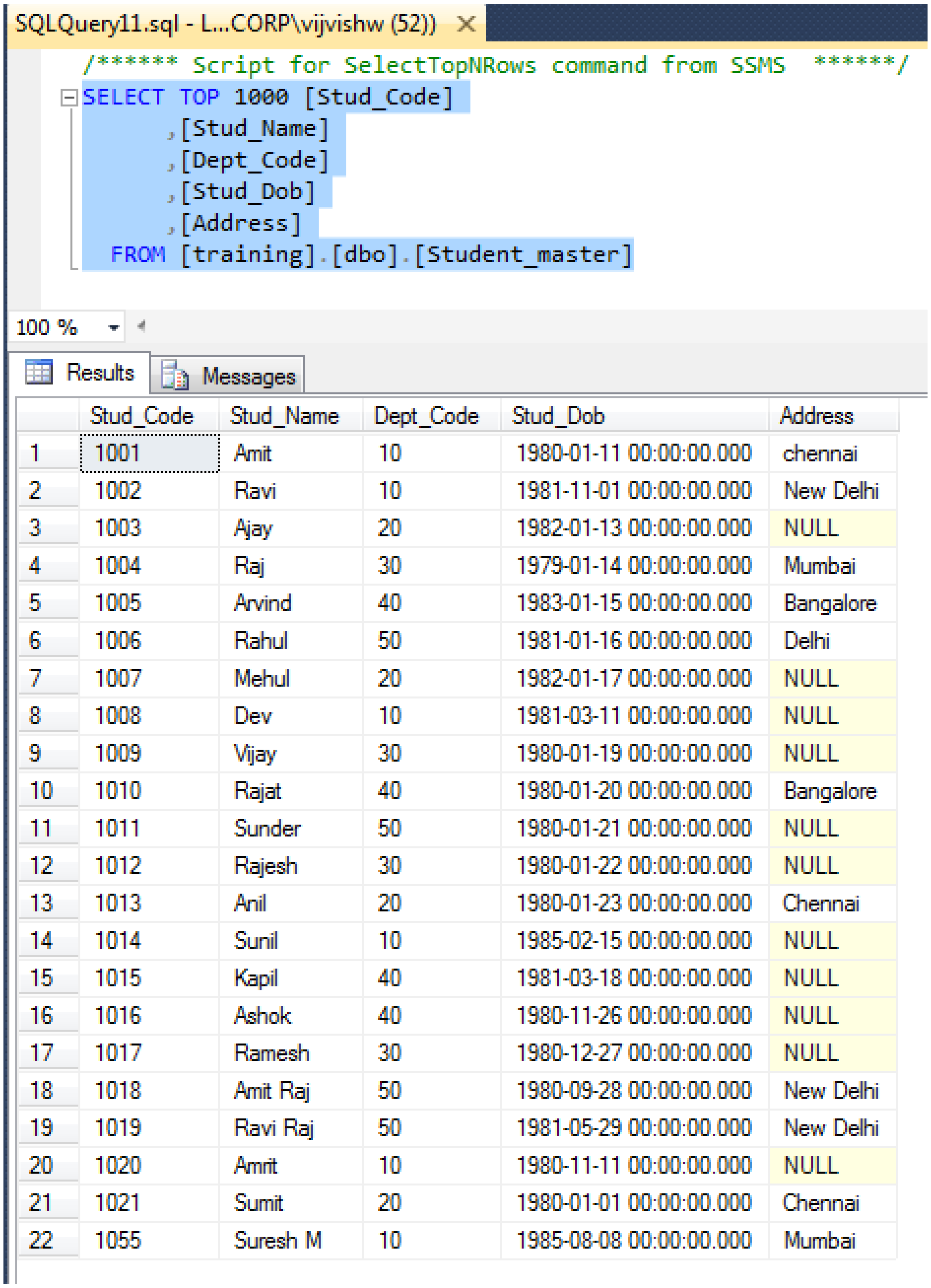


```
static void Main(string[] args)
{
    //Intializing Object context
    trainingEntities context = new trainingEntities();

    //Initializing a student object
    Student_master student = new Student_master
    {
        Stud_Code=1055,
        Stud_Name="Suresh M",
        Dept_Code=10,
        Stud_Dob=Convert.ToDateTime("08/08/1985"),
        Address="Mumbai"
    };

    //Adding the student object to EntitySet
    context.Student_master.Add(student);

    //Saving Chnages to Database
    context.SaveChanges();
    Console.WriteLine("Student Data Saved Successfully");
    Console.ReadLine();
}
```

Output :-



```
Student Data Saved Successfully
```



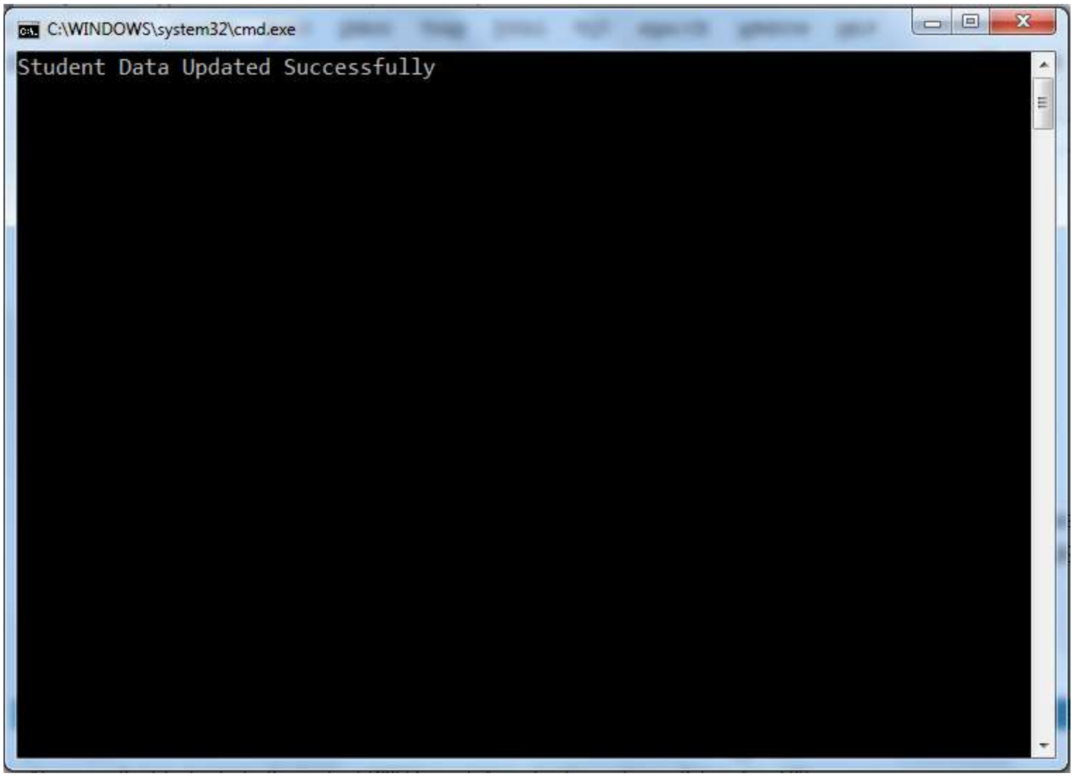
```sql
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [Stud_Code]
      ,[Stud_Name]
      ,[Dept_Code]
      ,[Stud_Dob]
      ,[Address]
  FROM [training].[dbo].[Student_master]
```

	Stud_Code	Stud_Name	Dept_Code	Stud_Dob	Address
1	1001	Amit	10	1980-01-11 00:00:00.000	chennai
2	1002	Ravi	10	1981-11-01 00:00:00.000	New Delhi
3	1003	Ajay	20	1982-01-13 00:00:00.000	NULL
4	1004	Raj	30	1979-01-14 00:00:00.000	Mumbai
5	1005	Arvind	40	1983-01-15 00:00:00.000	Bangalore
6	1006	Rahul	50	1981-01-16 00:00:00.000	Delhi
7	1007	Mehul	20	1982-01-17 00:00:00.000	NULL
8	1008	Dev	10	1981-03-11 00:00:00.000	NULL
9	1009	Vijay	30	1980-01-19 00:00:00.000	NULL
10	1010	Rajat	40	1980-01-20 00:00:00.000	Bangalore
11	1011	Sunder	50	1980-01-21 00:00:00.000	NULL
12	1012	Rajesh	30	1980-01-22 00:00:00.000	NULL
13	1013	Anil	20	1980-01-23 00:00:00.000	Chennai
14	1014	Sunil	10	1985-02-15 00:00:00.000	NULL
15	1015	Kapil	40	1981-03-18 00:00:00.000	NULL
16	1016	Ashok	40	1980-11-26 00:00:00.000	NULL
17	1017	Ramesh	30	1980-12-27 00:00:00.000	NULL
18	1018	Amit Raj	50	1980-09-28 00:00:00.000	New Delhi
19	1019	Ravi Raj	50	1981-05-29 00:00:00.000	New Delhi
20	1020	Amrit	10	1980-11-11 00:00:00.000	NULL
21	1021	Sumit	20	1980-01-01 00:00:00.000	Chennai
22	1055	Suresh M	10	1985-08-08 00:00:00.000	Mumbai

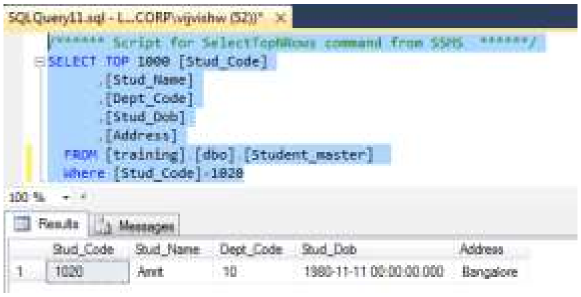
To Update a Record Add the following code

```
static void Main(string[] args)
{
    //Intializing Object context
    trainingEntities context = new trainingEntities();

    //Acquiring the Object which need to be updated
    Student_master student = (from s in context.Student_master.Where
                                (s => s.Stud_Code == 1020)
                              select s).FirstOrDefault();

    if (student != null)
    {
        student.Address = "Bangalore";
        context.SaveChanges();
        Console.WriteLine("Student Data Updated Successfully");
    }
    else
    {
        Console.WriteLine("Cannot Update Student\nStudent Not available");
    }
}
```

Output :-



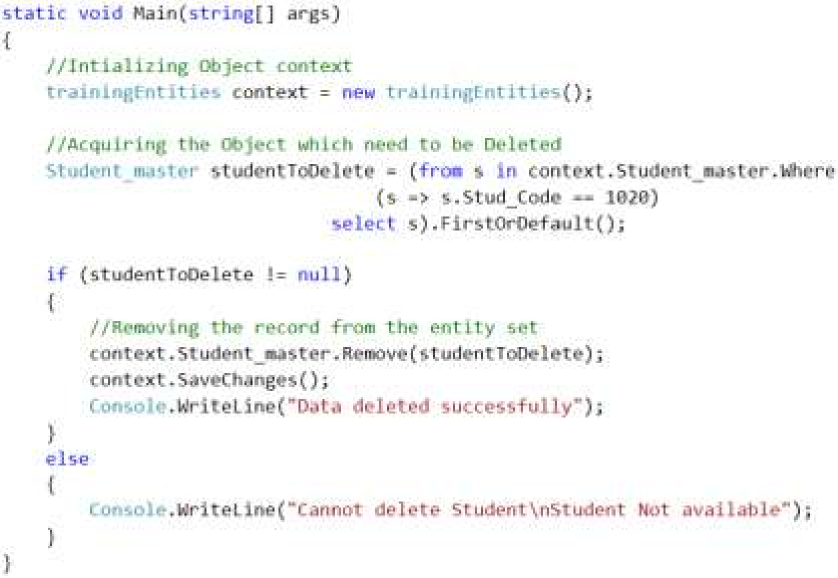
```
Student Data Updated Successfully
```



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [Stud_Code]
      ,[Stud_Name]
      ,[Dept_Code]
      ,[Stud_Dob]
      ,[Address]
  FROM [training].[dbo].[Student_master]
  where [Stud_Code]=1020
```

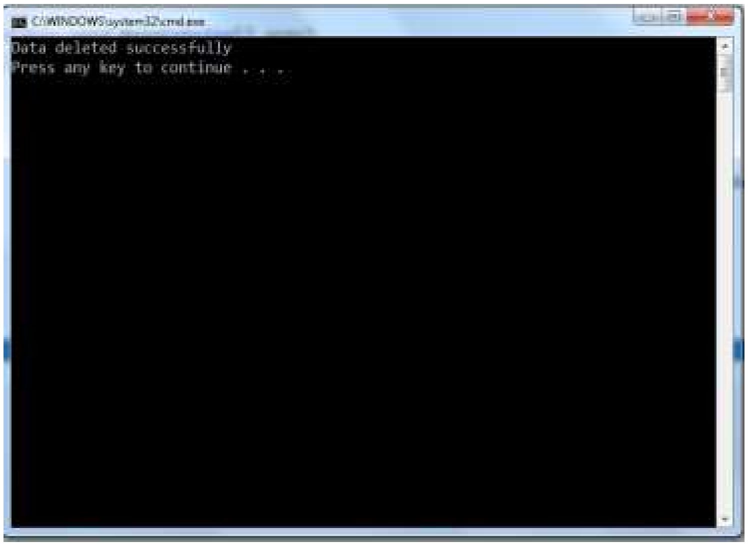
	Stud_Code	Stud_Name	Dept_Code	Stud_Dob	Address
1	1020	Amit	10	1980-11-11 00:00:00.000	Bangalore

To Delete a Record Add the following code

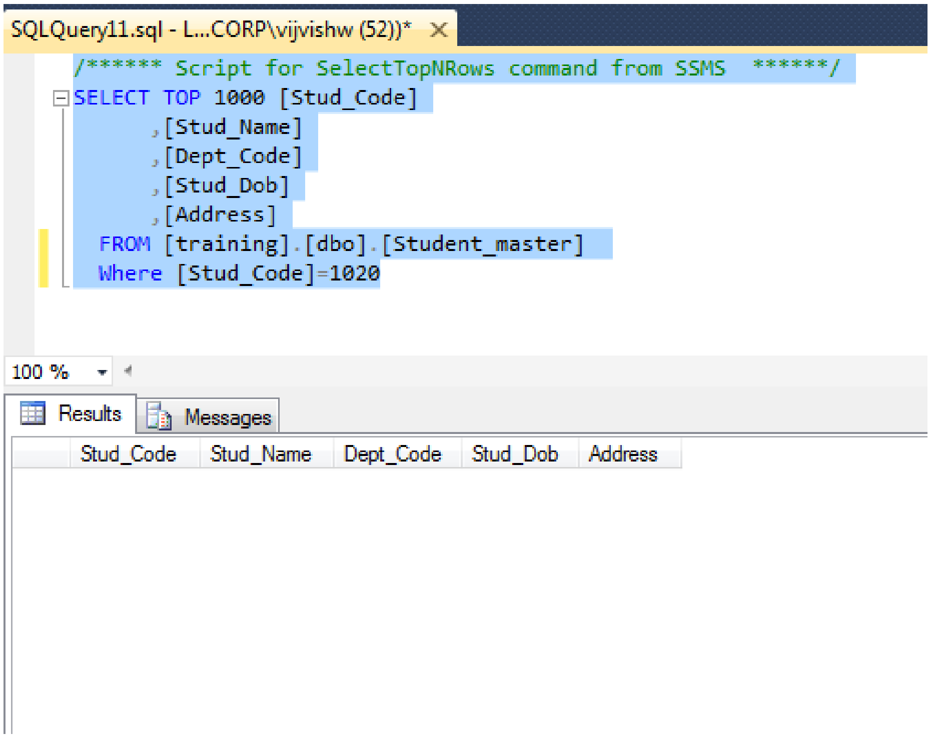


```
static void Main(string[] args)
{
    //Intializing Object context
    trainingEntities context = new trainingEntities();

    //Acquiring the Object which need to be Deleted
    Student_master studentToDelete = (from s in context.Student_master.Where
                                          (s => s.Stud_Code == 1020)
                                      select s).FirstOrDefault();

    if (studentToDelete != null)
    {
        //Removing the record from the entity set
        context.Student_master.Remove(studentToDelete);
        context.SaveChanges();
        Console.WriteLine("Data deleted successfully");
    }
    else
    {
        Console.WriteLine("Cannot delete Student\nStudent Not available");
    }
}
```

Output:-



```
Data deleted successfully
Press any key to continue . . .
```



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [Stud_Code]
      ,[Stud_Name]
      ,[Dept_Code]
      ,[Stud_Dob]
      ,[Address]
  FROM [training].[dbo].[Student_master]
  Where [Stud_Code]=1020
```

Stud_Code	Stud_Name	Dept_Code	Stud_Dob	Address

To-Do Assignments

1. Create a Console application to perform CRUD operation . You have to perform following step.
    a. Create a table named Employee which will have the following fields
        i. ID
        ii. Name
        iii. DOB
        iv. DOJ
        v. Designation
        vi. Salary
    b. Add a Entity Data Model to project which will include above mentioned Entity.
    c. Using LINQ to Entities write the following functionality and execute them
        i. Add a Employee details
        ii. Updating a Employee details
        iii. Searching for an Employee based on It ID
        iv. Deleteing an employee.