

# Assignment8

## QUESTION 1 – LINQ to Array / List Sorting (Ascending)

```
using System;
using System.Collections.Generic;
using System.Linq;

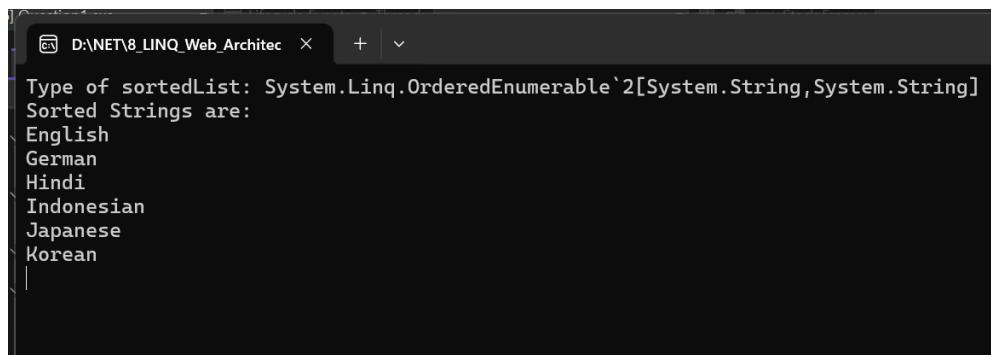
namespace Assignment8_Q1
{
    class Program
    {
        static void Main(string[] args)
        {
            List<string> languages = new List<string>
            {
                "Indonesian", "Korean", "Japanese", "English", "German", "Hindi"
            };

            var sortedList = from lang in languages
                            orderby lang
                            select lang;

            // Print type of sortedList
            Console.WriteLine("Type of sortedList: " + sortedList.GetType());

            Console.WriteLine("Sorted Strings are:");
            foreach (var item in sortedList)
            {
                Console.WriteLine(item);
            }

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



```
D:\NET\8_LINQ_Web_Architec > Type of sortedList: System.Linq.OrderedEnumerable`2[System.String,System.String]
Sorted Strings are:
English
German
Hindi
Indonesian
Japanese
Korean
```

👉 LINQ is a set of **extension methods**

👉 When we write using System.Linq;, we add methods like **OrderBy**, **Where**, **Select** to all types that implement **IEnumerable<T>**.

We unlock over **100 extension methods**, including: **Where**, **Select**, **OrderBy**, **ThenBy**, **GroupBy**, **Count**, etc.

## QUESTION 2 – LINQ Filter Students by Grade

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Assignment8_Q2
{
    class Student
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Grade { get; set; }
        public int GradePoint { get; set; }
    }

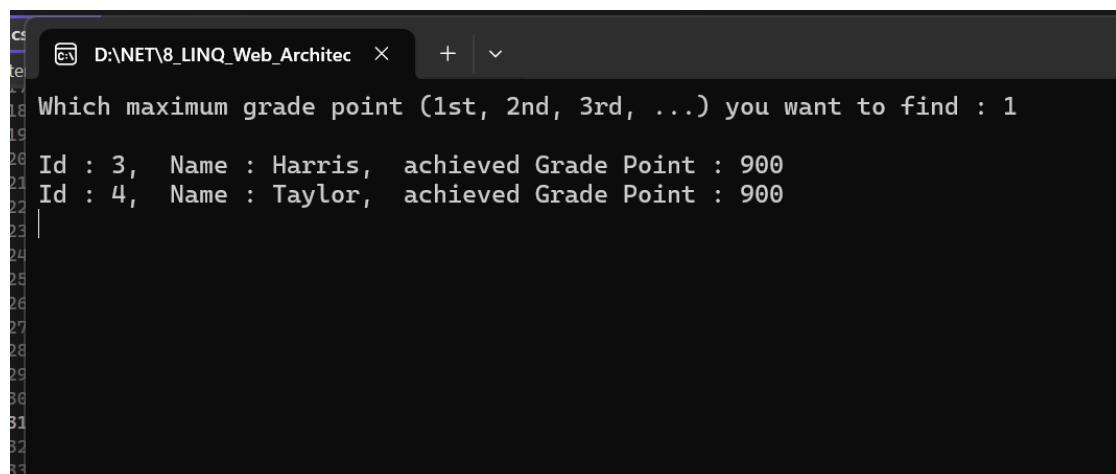
    class Program
    {
        static void Main(string[] args)
        {
            List<Student> students = new List<Student>()
            {
                new Student { Id = 1, Name = "John", Grade = 3, GradePoint = 850 },
                new Student { Id = 2, Name = "Emily", Grade = 2, GradePoint = 760 },
                new Student { Id = 3, Name = "Harris", Grade = 1, GradePoint = 900 },
                new Student { Id = 4, Name = "Taylor", Grade = 1, GradePoint = 900 },
                new Student { Id = 5, Name = "Chris", Grade = 3, GradePoint = 780 }
            };

            Console.WriteLine("Which maximum grade point (1st, 2nd, 3rd, ...) you want to find : ");
            int userGrade = int.Parse(Console.ReadLine());

            var result = from s in students
                        where s.Grade == userGrade
                        select s;

            Console.WriteLine();
            foreach (var s in result)
            {
                Console.WriteLine($"Id : {s.Id}, Name : {s.Name}, achieved Grade Point : {s.GradePoint}");
            }

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



```
cs D:\NET\8_LINQ_Web_Architec × + ▾
18 Which maximum grade point (1st, 2nd, 3rd, ...) you want to find : 1
19
20 Id : 3, Name : Harris, achieved Grade Point : 900
21 Id : 4, Name : Taylor, achieved Grade Point : 900
22
23
24
25
26
27
28
29
30
31
32
33
```

### QUESTION 3 — Frequency of Characters (Solved in C# with LINQ)

```
using System;
using System.Linq;

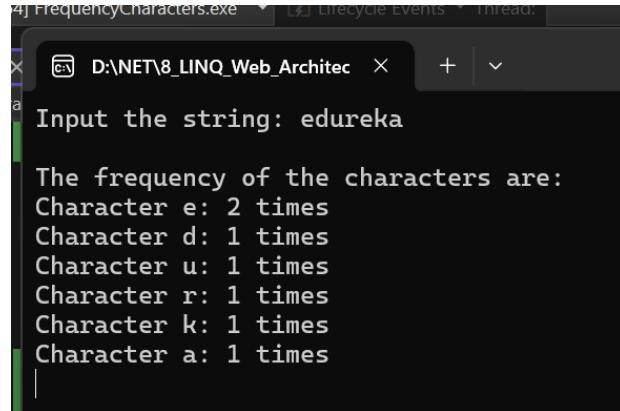
namespace Assignment8_Q3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Input the string: ");
            string input = Console.ReadLine();

            Console.WriteLine("\nThe frequency of the characters are:");

            var result =
                from c in input
                group c by c into g
                select new
                {
                    Character = g.Key,
                    Count = g.Count()
                };

            foreach (var item in result)
            {
                Console.WriteLine($"Character {item.Character}: {item.Count} times");
            }

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



The frequency of the characters are:

Character e: 2 times

Character d: 1 times

Character u: 1 times

Character r: 1 times

Character k: 1 times

Character a: 1 times

#### QUESTION 4 — Square of Numbers > 20 (Classic LINQ)

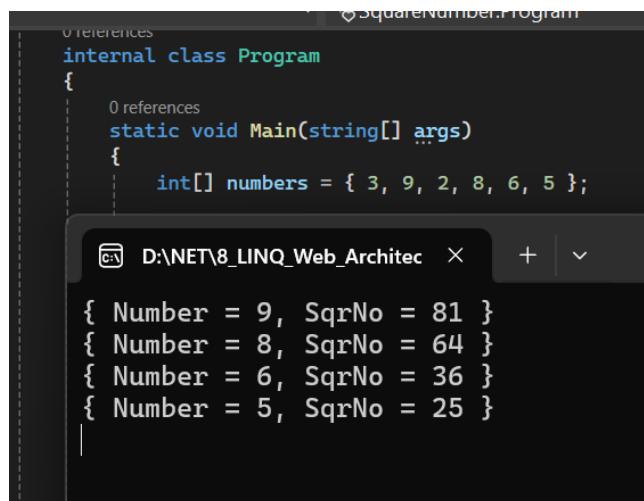
```
using System;
using System.Linq;

namespace Assignment8_Q4
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] numbers = { 3, 9, 2, 8, 6, 5 };

            var result =
                from n in numbers
                let sq = n * n
                where sq > 20
                select new
                {
                    Number = n,
                    SqrNo = sq
                };

            foreach (var item in result)
            {
                Console.WriteLine($"{{ Number = {item.Number}, SqrNo = {item.SqrNo} }}");
            }

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



The screenshot shows a .NET IDE interface with the code editor open. The code is identical to the one provided above. A modal window titled 'D:\NET\8\_LINQ\_Web\_Architec' is displayed, showing the output of the program. The output consists of four lines of text, each representing an anonymous object with 'Number' and 'SqrNo' properties:

```
{ Number = 9, SqrNo = 81 }
{ Number = 8, SqrNo = 64 }
{ Number = 6, SqrNo = 36 }
{ Number = 5, SqrNo = 25 }
```

## QUESTION 5 – Searching List Elements Starting With 'A'

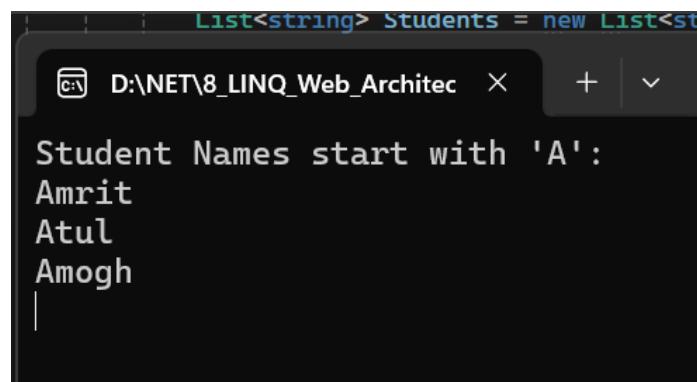
```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Assignment8_Q5
{
    class Program
    {
        static void Main(string[] args)
        {
            List<string> Students = new List<string>();
            Students.Add("Amrit");
            Students.Add("Sumit");
            Students.Add("Atul");
            Students.Add("Shaurya");
            Students.Add("Amogh");

            // Classic LINQ syntax
            var result =
                from s in Students
                where s.StartsWith("A", StringComparison.OrdinalIgnoreCase)
                select s;

            Console.WriteLine("Student Names start with 'A':");
            foreach (var name in result)
            {
                Console.WriteLine(name);
            }

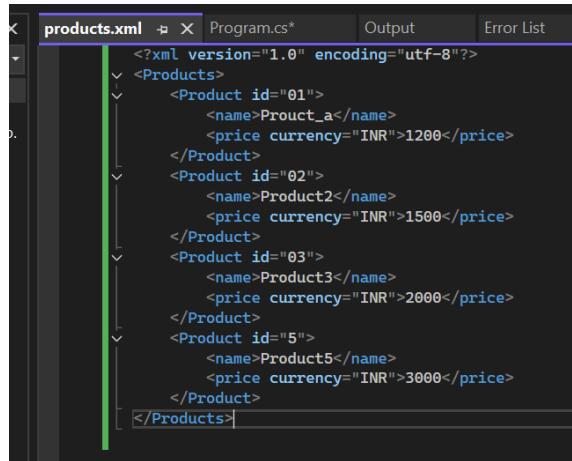
            Console.ReadLine();
        }
    }
}
```



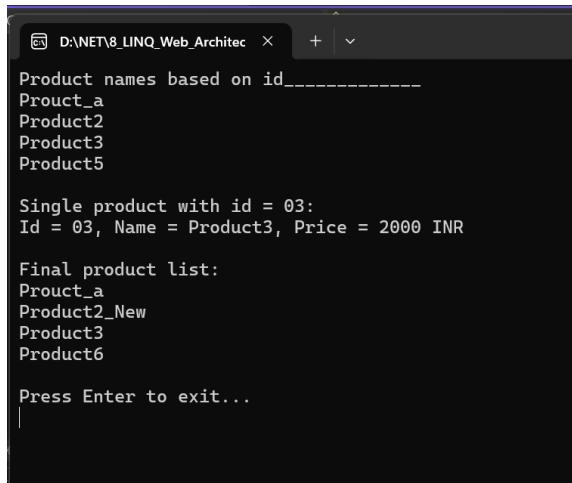
The screenshot shows a terminal window titled 'D:\NET\8\_LINQ\_Web\_Architec'. The output of the program is displayed, starting with the message 'Student Names start with 'A':'. Below this, four names are listed: 'Amrit', 'Atul', 'Shaurya', and 'Amogh'. The terminal window has a dark background and light-colored text.

```
List<string> Students = new List<string>();
D:\NET\8_LINQ_Web_Architec  X  +  ▾
Student Names start with 'A':
Amrit
Atul
Shaurya
Amogh
```

## Question6 CRUD with LINQ to XML



```
<?xml version="1.0" encoding="utf-8"?>
<Products>
    <Product id="01">
        <name>Product_a</name>
        <price currency="INR">1200</price>
    </Product>
    <Product id="02">
        <name>Product2</name>
        <price currency="INR">1500</price>
    </Product>
    <Product id="03">
        <name>Product3</name>
        <price currency="INR">2000</price>
    </Product>
    <Product id="05">
        <name>Product5</name>
        <price currency="INR">3000</price>
    </Product>
</Products>
```

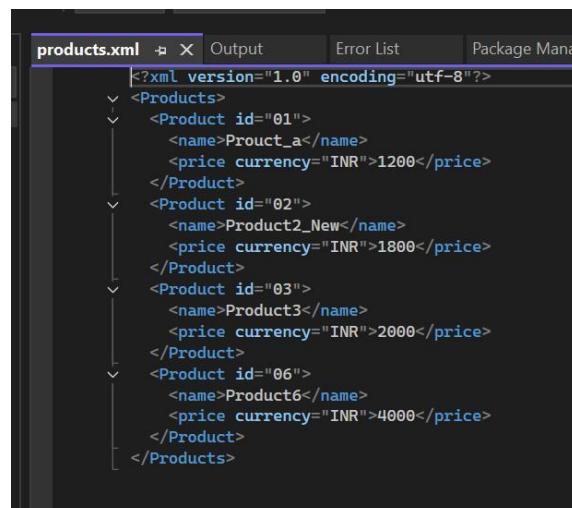


```
D:\NET\8_LINQ_Web_Architec> Product names based on id-----
Product_a
Product2
Product3
Product5

Single product with id = 03:
Id = 03, Name = Product3, Price = 2000 INR

Final product list:
Product_a
Product2_New
Product3
Product6

Press Enter to exit...|
```



```
<?xml version="1.0" encoding="utf-8"?>
<Products>
    <Product id="01">
        <name>Product_a</name>
        <price currency="INR">1200</price>
    </Product>
    <Product id="02">
        <name>Product2_New</name>
        <price currency="INR">1800</price>
    </Product>
    <Product id="03">
        <name>Product3</name>
        <price currency="INR">2000</price>
    </Product>
    <Product id="06">
        <name>Product6</name>
        <price currency="INR">4000</price>
    </Product>
</Products>
```

```

using System;
using System.Linq;
using System.Xml.Linq;

namespace Assignment8_Q6
{
    class Program
    {
        // Change this path if needed
        private const string XmlPath = "products.xml";

        static void Main(string[] args)
        {
            // Load the XML file
            XDocument doc = XDocument.Load(XmlPath);

            Console.WriteLine("Product names based on id_____");

            // READ ALL
            PrintAllProductNames(doc);

            //--- EXAMPLES OF CRUD OPERATIONS ---

            // ADD a new product
            AddProduct(doc, "06", "Product6", 4000, "INR");

            // UPDATE product name & price by id
            UpdateProduct(doc, "02", "Product2_New", 1800);

            // DELETE product by id
            DeleteProduct(doc, "5");

            // READ one product by id
            Console.WriteLine("\nSingle product with id = 03:");
            ReadOneProductById(doc, "03");

            // Save back to the file (optional)
            doc.Save(XmlPath);

            Console.WriteLine("\nFinal product list:");
            PrintAllProductNames(doc);

            Console.WriteLine("\nPress Enter to exit...");
            Console.ReadLine();
        }

        // READ ALL: show names (like sample output)
        private static void PrintAllProductNames(XDocument doc)
        {
            var products =
                from p in doc.Descendants("Product")
                select new
                {
                    Id = (string)p.Attribute("id"),
                    Name = (string)p.Element("name")
                };

            foreach (var p in products)
            {
                Console.WriteLine(p.Name);
            }
        }

        // ADD: add new <Product> element
        private static void AddProduct(XDocument doc, string id, string name, decimal price, string currency)
        {

```

```

XElement newProduct =
    new XElement("Product",
        new XAttribute("id", id),
        new XElement("name", name),
        new XElement("price",
            new XAttribute("currency", currency),
            price.ToString()))
    );

    doc.Root.Add(newProduct);
}

// DELETE: remove product with matching id
private static void DeleteProduct(XDocument doc, string id)
{
    var product =
        (from p in doc.Descendants("Product")
         where (string)p.Attribute("id") == id
         select p).FirstOrDefault();

    if (product != null)
    {
        product.Remove();
    }
}

// UPDATE: change name and price for product with given id
private static void UpdateProduct(XDocument doc, string id, string newName, decimal newPrice)
{
    var product =
        (from p in doc.Descendants("Product")
         where (string)p.Attribute("id") == id
         select p).FirstOrDefault();

    if (product != null)
    {
        product.Element("name")!.Value = newName;
        product.Element("price")!.Value = newPrice.ToString();
    }
}

// READ ONE: print a single product by id
private static void ReadOneProductById(XDocument doc, string id)
{
    var product =
        (from p in doc.Descendants("Product")
         where (string)p.Attribute("id") == id
         select p).FirstOrDefault();

    if (product != null)
    {
        string name = (string)product.Element("name");
        string price = (string)product.Element("price");
        string currency = (string)product.Element("price")?.Attribute("currency");

        Console.WriteLine($"Id = {id}, Name = {name}, Price = {price} {currency}");
    }
    else
    {
        Console.WriteLine($"Product with id = {id} not found.");
    }
}
}

```

## QUESTION 7 : Employee record Web application with XML File

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml.Linq;

namespace EmployeeWebApplicationXML
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                // Load the XML file
                XElement doc = XElement.Load(Server.MapPath("Employees.xml"));

                var data =
                    from emp in doc.Descendants("employee")
                    select new
                    {
                        EmpId = (int)emp.Element("empid"),
                        EmpName = (string)emp.Element("empname"),
                        Salary = (int)emp.Element("salary"),
                        Gender = (string)emp.Element("gender")
                    };
            }

            GridView1.DataSource = data.ToList();
            GridView1.DataBind();
        }
    }
}
```

EmpId	EmpName	Salary	Gender
1	Akshay	10000	Female
2	Shalu	20000	Female
3	Akki	30000	Male
4	Sateesh	50000	Male

The screenshot shows the Visual Studio 2022 IDE interface. The main area displays the XML file `employee.xml` with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<Employees>
    <employee>
        <empid>101</empid>
        <ename>Akshay</ename>
        <salary>10000</salary>
        <gender>Male</gender>
    </employee>

    <employee>
        <empid>102</empid>
        <ename>Vinay</ename>
        <salary>25000</salary>
        <gender>Male</gender>
    </employee>

    <employee>
        <empid>103</empid>
        <ename>Stella</ename>
        <salary>15000</salary>
        <gender>Female</gender>
    </employee>

    <employee>
        <empid>104</empid>
    </employee>

```

The Solution Explorer pane shows the project `EmployeeWebApplicationXML` with files `Connected Services`, `Properties`, `References`, `packages.config`, `Web.config`, and `WebForm1.aspx`. The Properties pane indicates the XML Document is in **Unicode (UTF-8)** encoding.

## QUESTION 8 : LINQ to SQL

SQLQuery2.sql - L...MTF1DV\akram (69)    SQLQuery1.sql - L...MTF1DV\akram (65)\*

```

CREATE TABLE CollegeTable
(
    RollNo      INT      PRIMARY KEY,
    Name        NVARCHAR(50) NOT NULL,
    FathersName NVARCHAR(50) NOT NULL,
    MothersName NVARCHAR(50) NOT NULL
);

INSERT INTO CollegeTable (RollNo, Name, FathersName, MothersName) VALUES
(1, 'Sursh', 'Ramesh Tyagi', 'Amrita'),
(2, 'Nikita', 'Amar Sharma', 'Sulekha'),
(3, 'Amogh', 'Suresh Avasthi', 'Manorama');

```

SQLQuery2.sql - L...MTF1DV\akram (69)    SQLQuery1.sql - L...MTF1DV\akram

```

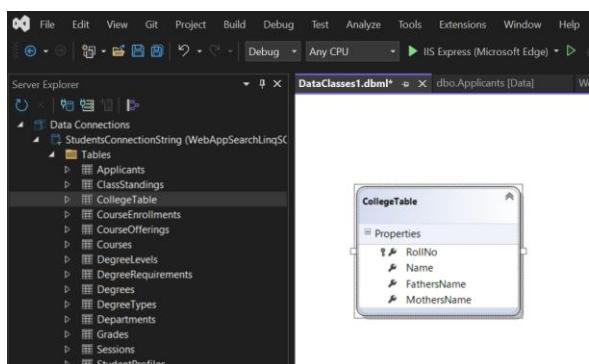
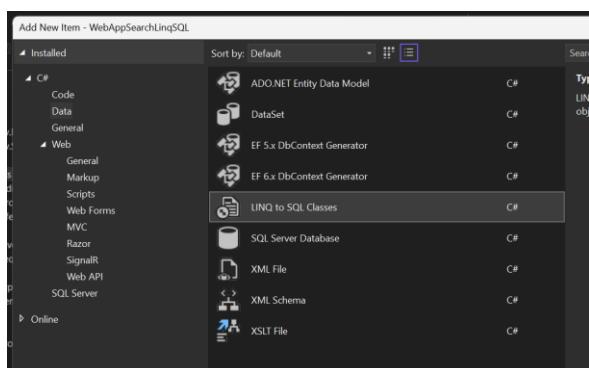
SELECT TOP (1000) [RollNo]
    , [Name]
    , [FathersName]
    , [MothersName]
FROM [Students].[dbo].[CollegeTable]

```

00 %

Results    Messages

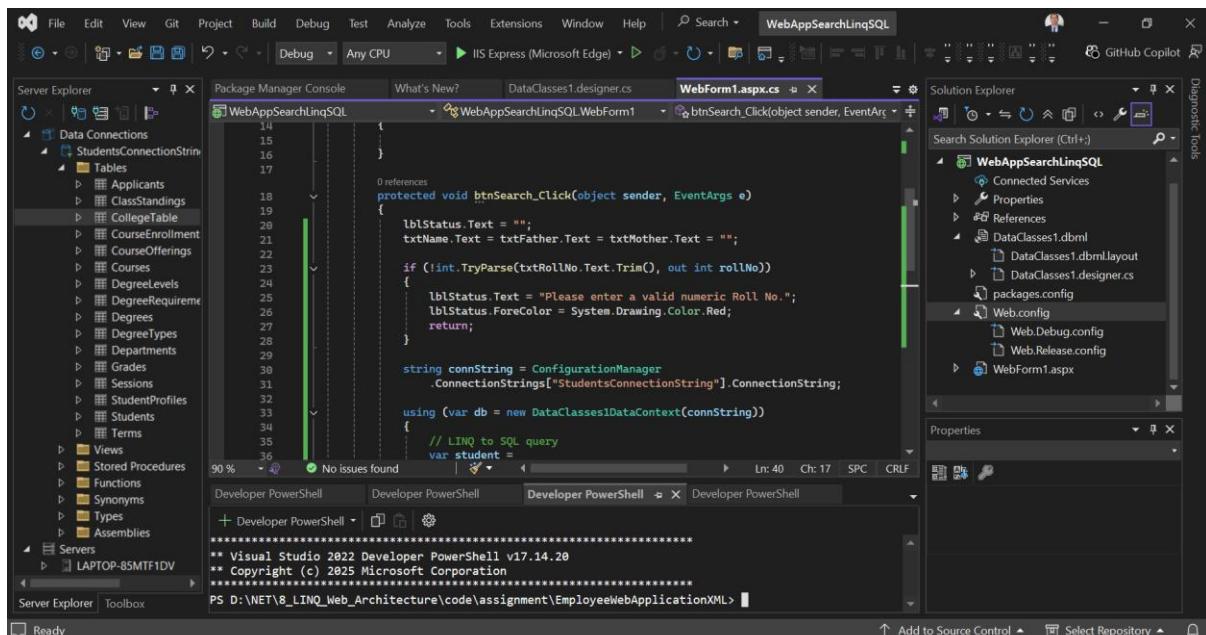
	RollNo	Name	FathersName	MothersName
1	1	Sursh	Ramesh Tyagi	Amrita
2	2	Nikita	Amar Sharma	Sulekha
3	3	Amogh	Suresh Avasthi	Manorama



```
xml version='1.0' encoding='utf-8'?>
<!--
For more information on how to configure your ASP.NET application, please visit
https://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <connectionStrings>
    <add name="StudentsConnectionString" connectionString="Data Source=LAPTOP-85MTF1DV;Initial Catalog=Students; providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.8" />
    <httpRuntime targetFramework="4.8" />
  </system.web>
  <system.codedom>
    <compilers>
      <compiler language="c#;cs;csharp" extension=".cs" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
      <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
    </compilers>
  </system.codedom>
</configuration>
```

A screenshot of a web browser window. The address bar shows the URL <https://localhost:44384/WebForm1.aspx>. The page title is "LINQ TO SQL". The form contains four text input fields: "Student Roll No" with value "1", "Student Name" with value "Sursh", "Father's Name" with value "Ramesh Tyagi", and "Mother's Name" with value "Amrita". Below the form, a green message says "Record found successfully."

A screenshot of a web browser window. The address bar shows the URL <https://localhost:44384/WebForm1.aspx>. The page title is "LINQ TO SQL". The form contains four text input fields: "Student Roll No" with value "111", "Student Name" (empty), "Father's Name" (empty), and "Mother's Name" (empty). Below the form, a red message says "No record found."



```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebAppSearchLinqSQL
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnSearch_Click(object sender, EventArgs e)
        {
            lblStatus.Text = "";
            txtName.Text = txtFather.Text = txtMother.Text = "";

            if (!int.TryParse(txtRollNo.Text.Trim(), out int rollNo))
            {
                lblStatus.Text = "Please enter a valid numeric Roll No.";
                lblStatus.ForeColor = System.Drawing.Color.Red;
                return;
            }

            string connString = ConfigurationManager
                .ConnectionStrings["StudentsConnectionString"].ConnectionString;

            using (var db = new DataClasses1DataContext(connString))
            {
                // LINQ to SQL query
                var student =
                    (from s in db.CollegeTables
                     where s.RollNo == rollNo
                     select s).FirstOrDefault();
            }
        }
    }
}

```

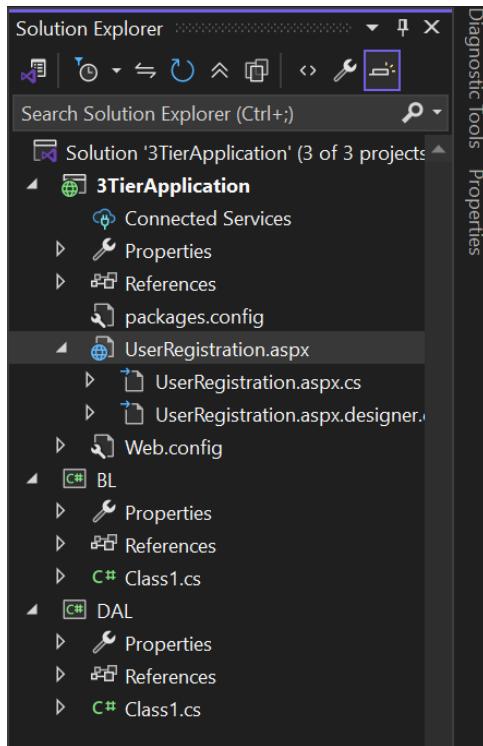
```
select s).SingleOrDefault();  
  
if (student == null)  
{  
    lblStatus.Text = "No record found. ";  
    lblStatus.ForeColor = System.Drawing.Color.Red;  
}  
else  
{  
    txtName.Text = student.Name;  
    txtFather.Text = student.FathersName;  
    txtMother.Text = student.MothersName;  
  
    lblStatus.Text = "Record found successfully. ";  
    lblStatus.ForeColor = System.Drawing.Color.Green;  
}  
}  
}  
}  
}
```

## QUESTION 9 : 3-tier web application

The screenshot shows the Microsoft Visual Studio interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, SQL, Test, Analyze, Tools, Extensions, Window, Help, and a search icon. The left pane is the Server Explorer, displaying a tree view of Data Connections, Tables, Views, and Stored Procedures under 'laptop-85mtf1dv.Employ'. The right pane is the Object Explorer, showing the database 'dbo.user101 [Data]' with tables 'UserRegistration.aspx.cs', 'Class1.cs', and 'UserRegistration.aspx'. A preview window displays a table with columns id, Name, City, and email, containing three rows of data.

The screenshot shows the Microsoft Visual Studio interface with multiple windows open. The Server Explorer pane shows the same database structure as the previous screenshot. The Solution Explorer pane shows a solution named '3TierApplication' with projects '3TierApplication', 'BL', and 'DAL'. The Design view pane shows a form titled '3-Tier Architecture' with fields for Name, City, and Email, and a save button. Below the form is a table with columns 'Column0', 'Column1', and 'Column2', containing several rows of data. The bottom pane shows the code editor for 'Class1.cs' with the namespace 'BL'.

The screenshot shows a web browser window with the URL 'localhost:44308/UserRegistration.aspx'. The page title is '3-Tier Architecture'. It contains a form with fields for Name ('Tom Cat'), City ('Mars'), and Email ('tom.cat@gmail.com'). Below the form is a save button. A table at the bottom shows the same data as the previous screenshots, with columns 'id', 'Name', 'City', and 'email', and two rows of data: (1, John Smith, Lausannegeles, John.Smith@hotmail.com) and (2, Tom Cat, Mars, tom.cat@gmail.com).



## BL Business Logic

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

namespace BL
{
    public class Class1
    {

    }

    public class BLL {
        DL ob1 = new DL();
        public void Inputuser(string name, string city, string email)
        {
            ob1.insertData(name, city, email);
        }

        public object selectUser()
        {
            return ob1.selectData();
        }

    }
}
```

## DAL Data Access Layer

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

namespace DAL
{
    public class Class1
    {

    }

    public class DL
    {
        SqlConnection con = new SqlConnection(
            @"Data Source=LAPTOP-85MTF1DV;
            Initial Catalog=Employees;
            Integrated Security=True;
            TrustServerCertificate=True");

        public void insertData(string name, string city, string email)
        {
            SqlDataAdapter sda = new SqlDataAdapter("insert into user101 values('"+name+"','"+city+"','"+email+"')", con);
            DataTable dt = new DataTable();
            sda.Fill(dt);

        }

        public object selectData()
        {
            SqlDataAdapter sda = new SqlDataAdapter("select * from user101", con);
            DataTable dt = new DataTable();
            sda.Fill(dt);
            return dt;

        }
    }
}
```

## View Layer

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using BL;

namespace _3TierApplication
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
```

```

{
}

protected void Button1_Click(object sender, EventArgs e)
{
    BLL ob2 = new BLL();
    ob2.Inputuser(TextBox1.Text, TextBox2.Text, TextBox3.Text);
    GridView1.DataSource = ob2.selectUser();
    GridView1.DataBind();
}
}

```

localhost:44308/UserRegistration.

https://localhost:44308/UserRegistration

3-Tier Architecture

Name	Akram MTir
City	Lausanne
Email	akram.mtirhotmail.com

**Save**

localhost:44308/UserRegistration.

https://localhost:44308/UserRegistration.aspx

3-Tier Architecture

Name	Akram MTir
City	Lausanne
Email	akram.mtirhotmail.com

<b>id</b>	<b>Name</b>	<b>City</b>	<b>email</b>
1	John Smith	Lausannegeles	John.Smith@hotmail.com
2	Tom Cat	Mars	tom.cat@gmail.com
3	Akram MTir	Lausanne	akram.mtirhotmail.com