Lab Exercise: Entity Framework Core-Code-First Approach with Single Entity "Student"

Objective:

To practice and understand the basics of Entity Framework (EF) Core using the Code-First approach. The lab will guide you through setting up EF Core, configuring it, creating a migration, and performing CRUD operations on a Student entity.

Lab 1: Initial EF Core Setup

Step 1: Create a New Project

1. Open Visual Studio (or your preferred IDE).
2. Create a new **Console Application** project.
3. Name the project StudentManagement.

Step 2: Add Entity Framework Core Packages

1. Right-click on the project and select **Manage NuGet Packages**.
2. Install the following packages:
    - Microsoft.EntityFrameworkCore
    - Microsoft.EntityFrameworkCore.SqlServer (or any other database provider you prefer)
    - Microsoft.EntityFrameworkCore.Tools

Step 3: Create the Data Context

1. In the project, create a new folder named Data.
2. Inside the Data folder, create a class named StudentContext and define it as follows:

```
using Microsoft.EntityFrameworkCore;

namespace StudentManagement.Data
{
    public class StudentContext : DbContext
    {
        public DbSet<Student> Students { get; set; }
```

```
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {

optionsBuilder.UseSqlServer(@"Your_Connection_String_Here");
        }
    }
}
```

Step 4: Create the Student Entity

1. In the root of the project, create a class named Student:

```
namespace StudentManagement
{
    public class Student
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public string Email { get; set; }
    }
}
```

Lab 2: Configuring EF Core & Code-First Migrations

Step 1: Enable Migrations

1. Open the **Package Manager Console** in Visual Studio.
2. Run the following command to enable migrations:

```
Add-Migration InitialCreate
```

3. After the command is executed, a new folder named Migrations will be created with the necessary migration files.

Step 2: Apply the Migration

1. In the Package Manager Console, apply the migration to the database by running:

```
Update-Database
```

2. Ensure that the database is created with the Students table.

Lab 3: Performing CRUD Operations

Step 1: Create a Student Record

1. In the Program.cs file, modify the Main method to add a new student record:

```
using System;
using StudentManagement.Data;

namespace StudentManagement
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var context = new StudentContext())
            {
                var student = new Student
                {
                    Name = "John Doe",
                    Age = 20,
                    Email = "john.doe@example.com"
                };

                context.Students.Add(student);
                context.SaveChanges();
            }

            Console.WriteLine("Student record added.");
        }
    }
}
```

Step 2: Read Student Records

1. Extend the Main method to retrieve and display the student records:

```
using System.Linq;

// After adding the student record...
```

```
using (var context = new StudentContext())
{
    var students = context.Students.ToList();

    foreach (var student in students)
    {
        Console.WriteLine($"ID: {student.Id}, Name: {student.Name}, Age: {student.Age}, Email: {student.Email}");
    }
}
```

Step 3: Update a Student Record

1. Add code to update an existing student record:

```
using (var context = new StudentContext())
{
    var student = context.Students.FirstOrDefault(s => s.Name == "John Doe");

    if (student != null)
    {
        student.Age = 21;
        context.SaveChanges();
        Console.WriteLine("Student record updated.");
    }
}
```

Step 4: Delete a Student Record

1. Finally, add code to delete a student record:

```
using (var context = new StudentContext())
{
    var student = context.Students.FirstOrDefault(s => s.Name == "John Doe");

    if (student != null)
    {
```

```
            context.Students.Remove(student);
            context.SaveChanges();
            Console.WriteLine("Student record deleted.");
        }
}
```

Lab Completion

- Verify that the CRUD operations work as expected by checking the database and the console outputs.
- Experiment by adding, updating, reading, and deleting more records to gain a deeper understanding of EF Core.