1. Entity Frameworkcore Codefirst approach
   a) store student information
   b) Fetch student information using LINQ

nuget packages: Entity Frameworkcore, Entity Frameworkcore.Design, Entity Frameworkcore.SqlServer, Entity Frameworkcore.Tools
Nuget Console Commands: Add-Migration Initial, update-database(Run after model and context files created and also database created)

model.cs

```
public class Student
{
   public int StudentID { get; set; }
   public string StudentName { get; set; }

}
```

context.cs

```
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics.Metrics;
using System;

public class SchoolContext : DbContext
{
   protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
   {
      optionsBuilder.UseSqlServer(@"Data Source = (localdb)\MSSQLLocalDB;
Database=StudentDatabase;Initial Catalog = master; Integrated Security = True; Connect
Timeout = 30; Encrypt = False; TrustServerCertificate = False; ApplicationIntent = ReadWrite;
MultiSubnetFailover = False");
   }
   public DbSet<Student> Students { get; set; }

}
```

program.cs

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

      using (var ctx = new SchoolContext())
      {
         Console.Write("Enter a name and id for a new student ");
         var name = Console.ReadLine();
         var stud = new Student() { StudentName = name };
```

```
            ctx.Students.Add(stud);
            ctx.SaveChanges();

            //fetching
            var query = from b in ctx.Students
                        select b;

            Console.WriteLine("All students in the database:");
            foreach (var item in query)
            {
               Console.WriteLine(item.StudentName);
            }
         }
      }
   }
```

2. Stored procedure Program

   nuget packages: Entity Frameworkcore, Entity Frameworkcore.Design, Entity Frameworkcore.SqlServer, Entity Frameworkcore.Tools
   Nuget Console Commands: Add-Migration spGetStudents, update-database(Run after code is placed in up method)

   spGetStudents.cs
   Place the below code inside up method

```
var sp = @"CREATE PROCEDURE [dbo].[GetStudents]
                @FirstName varchar(50)
            AS
            BEGIN
               SET NOCOUNT ON;
               select * from Students where StudentName like @FirstName +'%'
            END";

            migrationBuilder.Sql(sp);
```

   model.cs

```
public class Student
{
   public int StudentID { get; set; }
   public string StudentName { get; set; }

}
```

   context.cs

```
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics.Metrics;
```

```
using System;

public class SchoolContext : DbContext
{
   protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
   {
      optionsBuilder.UseSqlServer(@"Data Source = (localdb)\MSSQLLocalDB;
Database=StudentDatabase;Initial Catalog = master; Integrated Security = True; Connect
Timeout = 30; Encrypt = False; TrustServerCertificate = False; ApplicationIntent = ReadWrite;
MultiSubnetFailover = False");
   }
   public DbSet<Student> Students { get; set; }

}
```

program.cs

```
using Microsoft.EntityFrameworkCore;

class Program
{
   static void Main(string[] args)
   {

      using (var ctx = new SchoolContext())
      {

         var result = ctx.Students.FromSqlRaw("GetStudents J").ToList();

         Console.WriteLine("All students in the database:");
         foreach (var item in result)
         {
            Console.WriteLine(item.StudentName);
         }
      }
   }
}
```

3. Program to store student information using ADO. Net

```
using System;

using Microsoft.Data.SqlClient;

namespace ConsoleApp10
{
   internal class dbconnectivity
   {
      static void Main()
```

```
    {
        string connString = @"Data Source = DESKTOP-PSJ0L2I\SQLEXPRESS;Initial Catalog =
master; Integrated Security = True; TrustServerCertificate=True ";

        string SELECT_TABLE = @"select * from employee";
        SqlConnection conn = new SqlConnection(connString);

        try
        {
            conn.Open();

            SqlCommand cmd1 = new SqlCommand("select * from Students", conn);
            SqlDataReader dr = cmd1.ExecuteReader();
            while (dr.Read())
            {
                Console.WriteLine(dr[0]);
                Console.WriteLine(dr[1]);
                Console.WriteLine("");
            }
        }

        catch (Exception e)
        {
            Console.WriteLine("Error: " + e);
        }
        finally
        {
            conn.Close();
        }
        Console.ReadKey();
    }
  }

}
```