Mock ASSESSMENT 5a

OVERVIEW

Implement a SQL table and connect it to an MVC project using entity. This table will be employee table with various info on each employee. You will use this data to determine what benefits employees receive upon retiring.

SETUP

Create a new ASP.NET Core Web Application named **Assessment5Mock**.

BUILD SPECIFICATIONS

The assessment is worth ten points, one for each of the test cases below. Pay special attention to the items in bold. **You must use these in your program exactly, including capitalization**, in order to get the points. Note: You'll be graded on what's in the **HomeController,** but you are allowed to create another controller to test and experiment with your connection.

Within SSMS, using the following query make a database named **EmployeeDb** and a table named **Employee** with the following datatypes:

```
CREATE DATABASE EmployeeDB;
USE [EmployeeDb]
GO

CREATE TABLE [dbo].[Employee](
   [Id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
   [FirstName] [nvarchar](40) NOT NULL,
   [Age] [int] NOT NULL,
   [Salary] [money] NOT NULL
)
GO

INSERT INTO Employee ( FirstName, Age, Salary)
VALUES( 'Jimmy', 70, 90000 ),
   ('Sandy', 50, 45000),
   ('Allen', 25, 30000)
GO
```

Your table should have the following columns:

- **Id** (Primary Key) int, auto increment
- **FirstName**, nvarchar(40)
- **Age**, int
- **Salary** money

Id	FirstName	Age	Salary
1	Jimmy	70	90000
2	Sandy	50	45000
3	Allen	25	30000

Now build the app:

1. Using Entity Framework, bring the table into your **Assessment6Mock** project. Call the context **EmployeeDb**.

Your web application will have the **Employees** view which will display the following:

2. An HTML table which will display each Employee in the database, displaying FirstName, Age, and Salary. This data should be passed in as a list of Employee Models.
    a. Syntax for this:
        i. In ActionResult: return View(listHere)
        ii. In View: @model List<Employee>
3. A form on the bottom of the page that allows the user to select an employee by id.
    a. The form must have a single input named **Id** which will be of type number. The form allows the user to select an employee by Id
    b. The form submits to the ActionResult **RetirementInfo**.
    c.

4. Create a **Retirement** model with the following properties:
    a. Bool CanRetire
    b. Float Benefits

4. The action **RetirementInfo** will take in an int **Id** as a parameter. Do the following in the action:

    i. Find the employee from the database with the matching Id (Assume you will get valid inputs here)
    ii. Create an empty **Retirement** model, this will be passed down to the view
    iii. If the employee is older than 60 set **CanRetire** in the **Retirement** model to **true**
    iv. Else set **CanRetire** to **false**
    v. Next set **Benefits** in the **Retirement** model to be **60%** of the Employee's **Salary**

4. In the **RetirementInfo** view, take in the **Retirement** model you set up in the action, and display both the **Benefits** and **CanRetire** properties