Sql Query :

CREATE DATABASE StudentDb;

USE StudentDb;

CREATE TABLE StudentMarks (

StudentID INT PRIMARY KEY ,

StudentName VARCHAR(50) NOT NULL,

Subject VARCHAR(50) NOT NULL,

Marks INT NOT NULL

)

INSERT INTO StudentMarks VALUES (1,'Dhana', 'Math', 90)

INSERT INTO StudentMarks VALUES (2,'Lakshmi', 'Science', 80)

INSERT INTO StudentMarks VALUES (3,'Rani', 'English', 70)

SELECT AVG(Marks)AS AverageMarks,

MAX(Marks) AS MaximumMarks ,

MIN(Marks) AS MinimumMarks

FROM StudentMarks

SELECT \* FROM StudentMarks;

StudentMarks model class:

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace section7.Models

{

[Table("StudentMarks")]

public class StudentMarks

{

[Key]

public int StudentId { get; set; }

public string? StudentName { get; set; }

public string? Subject { get; set; }

public int Marks { get; set; }

}

}

StudentMarksControllers:

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using section7.Data;

using section7.Models;

namespace section7.Controllers

{

public class StudentMarksController : Controller

{

private readonly SchoolDbContext \_context;

public StudentMarksController(SchoolDbContext context)

{

\_context = context;

}

// GET: StudentMarks

public async Task<IActionResult> Index()

{

return \_context.StudentMarks != null ?

View(await \_context.StudentMarks.ToListAsync()) :

Problem("Entity set 'SchoolDbContext.StudentMarks' is null.");

}

// GET: StudentMarks/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null || \_context.StudentMarks == null)

{

return NotFound();

}

var studentMarks = await \_context.StudentMarks

.FirstOrDefaultAsync(m => m.StudentId == id);

if (studentMarks == null)

{

return NotFound();

}

return View(studentMarks);

}

// GET: StudentMarks/Create

public IActionResult Create()

{

return View();

}

// POST: StudentMarks/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("StudentId,StudentName,Subject,Marks")] StudentMarks studentMarks)

{

if (ModelState.IsValid)

{

\_context.Add(studentMarks);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(studentMarks);

}

// GET: StudentMarks/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null || \_context.StudentMarks == null)

{

return NotFound();

}

var studentMarks = await \_context.StudentMarks.FindAsync(id);

if (studentMarks == null)

{

return NotFound();

}

return View(studentMarks);

}

// POST: StudentMarks/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("StudentId,StudentName,Subject,Marks")] StudentMarks studentMarks)

{

if (id != studentMarks.StudentId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(studentMarks);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!StudentMarksExists(studentMarks.StudentId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(studentMarks);

}

// GET: StudentMarks/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null || \_context.StudentMarks == null)

{

return NotFound();

}

var studentMarks = await \_context.StudentMarks

.FirstOrDefaultAsync(m => m.StudentId == id);

if (studentMarks == null)

{

return NotFound();

}

return View(studentMarks);

}

// POST: StudentMarks/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (\_context.StudentMarks == null)

{

return Problem("Entity set 'SchoolDbContext.StudentMarks' is null.");

}

var studentMarks = await \_context.StudentMarks.FindAsync(id);

if (studentMarks != null)

{

\_context.StudentMarks.Remove(studentMarks);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool StudentMarksExists(int id)

{

return (\_context.StudentMarks?.Any(e => e.StudentId == id)).GetValueOrDefault();

}

}

}

Layout.cshtml:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - section7</title>

<**link** rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

<**link** rel="stylesheet" **href**="~/section7.styles.css" **asp-append-version**="true" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<**a** class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">section7</**a**>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Home</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="StudentMarks" **asp-action**="Index">StudentMarks</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</**a**>

</li>

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2023 - section7 - <**a** **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</**a**>

</div>

</footer>

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

<**script** **src**="~/js/site.js" **asp-append-version**="true"></**script**>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

Index.cshtml:

@model IEnumerable<section7.Models.StudentMarks>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<p>

<**a** **asp-action**="Create">Create New</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.StudentId)

</th>

<th>

@Html.DisplayNameFor(model => model.StudentName)

</th>

<th>

@Html.DisplayNameFor(model => model.Subject)

</th>

<th>

@Html.DisplayNameFor(model => model.Marks)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.StudentId)

</td>

<td>

@Html.DisplayFor(modelItem => item.StudentName)

</td>

<td>

@Html.DisplayFor(modelItem => item.Subject)

</td>

<td>

@Html.DisplayFor(modelItem => item.Marks)

</td>

<td>

<**a** **asp-action**="Edit" **asp-route-id**="@item.StudentId">Edit</**a**> |

<**a** **asp-action**="Details" **asp-route-id**="@item.StudentId">Details</**a**> |

<**a** **asp-action**="Delete" **asp-route-id**="@item.StudentId">Delete</**a**>

</td>

</tr>

}

</tbody>

Appsettings.Json:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"SchoolDbContext": "Server=LAPTOP-JL3U744Q;Database=StudentDb;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

}