|  |  |  |
|---|---|---|
| **Name: Kamran butt** | **Roll no:** 20101001-037 | **Section: Grey** |
| **Semester**:7th | **Instructor:** Sir Museb Khalid | **Title:** Assignment 2 |
| **Course: Advanced** Web Engineering |  | **Due Date:** 12-12-2023 |

1. **Retrieve Information from the Database:**

   a. **Write a LINQ query to get the names and room numbers for classes with more than 100 students.**

**var query = from c in class join e in enrolled on c.cid equals e.cid group c by c.cid into g where g.Count() > 100**
**select new { Name = g.First().name, Room = g.First().room_number };**

   b. **Write a LINQ query to get the ids and majors of students who take no classes with teachers in department 22.**

**var query = student.Join(enrolled, s => s.sid, e => e.sid, (s, e) => new { s, e })**

```
                  .Join(class, se => se.e.cid, c => c.cid,
(se, c) => new { se.s, se.e, c })
                  .Join(faculty, sec => sec.c.fid, f => f.fid,
(sec, f) => new { sec.s, sec.e, sec.c, f })
                  .GroupBy(secf => secf.s.sid)
                  .Where(g => !g.Any(secf =>
secf.f.deptid == 22))
                  .Select(g => new { Id = g.Key, Major =
g.First().s.major });
```

2. **Additional LINQ Challenges:**

**Choose any three of the following LINQ challenges and write appropriate queries:**

a. **Retrieve the names of students who are enrolled in classes and have not yet received any marks.**

```
var query = from s in student

            join e in enrolled on
s.sid equals e.sid            where
e.id == null
            select s.sname;
```

b. **Find the average age of students in each major. Display the major and the average age. var query = from s in student group s by s.major into g**

**select new { Major = g.Key, AverageAge = g.Average(s => s.age) };**

c. **Get the names of students who are enrolled in more than two classes. Display the student name and the number of classes they are enrolled in. var query = from s in student**

```
            join e in enrolled
on s.sid equals e.sid
group s by s.sid into g
where g.Count() > 2
            select new { Name = g.First().sname,
Classes = g.Count() };
```

**Model class**

**public class Student**

```
        {
            public int Id { get; set; }
            public string? Name { get; set; }
            public string? Major { get; set; }

        }
```

**Program .cs file**

```
builder.Services.AddDbContext<DbContext>
    (options =>
options.UseSqlServer(builder.Configuration.GetConnectionS
tring("Assign")))
```

**Dbcontext class**

```
public class Assignment2DbContext:DbContext
   {
       public
Assignment2DbContext(DbContextOptions<Assignment2DbCo
ntext> options) : base(options)
       { }
       public DbSet<Class> Class { get; set; }
       public DbSet<Student> Students { get; set; }
       public DbSet<Faculty> Facultys { get; set; }
       public DbSet<Enrolled> Enrolleds { get; set; }
```

```
    }

using Microsoft.EntityFrameworkCore;
namespace Assign2_linq.Data
{

    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext>
options) : base(options)
        {
        }
        public DbSet<Student> Students { get; set; }
    }
}

@page "/students/create"
@using Assign2_linq.Data
@inject AppDbContext DbContext
@inject NavigationManager navigationManager

<h3>Create Student</h3>

<EditForm Model="@newStudent"
OnValidSubmit="AddStudent">
```

```
    <DataAnnotationsValidator />
    <ValidationSummary />

    <div class="form-group">
        <label for="name">Name:</label>
        <InputText id="name" class="form-control" @bind-
Value="newStudent.Name" />
    </div> <div class="form-group">
        <label for="name">Name:</label>
        <InputText id="name" class="form-control" @bind-
Value="newStudent.Name" />
    </div>
    <div class="form-group">
        <label for="course">Course:</label>
        <InputText id="course" class="form-control" @bind-
Value="newStudent.Course" />
    </div>

    <button type="submit" class="btn btn-
primary">Save</button>
    <button><NavLink href="/students/delete">Delete
Student</NavLink></button>
    <button><NavLink href="/students/edit">Edit
Student</NavLink></button>
</EditForm>
```

```
@code {
    Student newStudent = new Student();
    string errorMessage;
@code {
    Student newStudent = new Student();
    string errorMessage;

    async Task AddStudent()
    {
        try
        {
            // Add the new student to the database
            DbContext.Students.Add(newStudent);
            await DbContext.SaveChangesAsync();

            // Navigate back to the student list
            navigationManager.NavigateTo("/students");
        }
        catch (Exception ex)
        {
            // Log the exception (you can use a logging framework
or simply print to console)
            Console.WriteLine($"Exception: {ex.Message}");
            // Set an error message to display to the user
            errorMessage = "An error occurred while saving the
student. Please try again.";
        }
    }
}
namespace Assign2_linq.Data
{
    public class Student
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Course { get; set; }
    }
}
```