**Exercise 5: Employee Management System - Defining Query Methods**

**Business Scenario:**

Enhance your repository to support custom queries.

**Instructions:**

1. **Defining Query Methods:**
   * Use keywords in method names to create custom query methods.
   * Implement custom query methods using the **@Query** annotation.

**EmployeeRepository.java🡪**

package com.code.employee.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import com.code.employee.entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer>{

    // Using the named query defined in Employee entity

    @Query(name = "Employee.findByEmail")

    Employee findByEmailNamed(@Param("email") String email);

    // Using the named query defined in Employee entity

    @Query(name = "Employee.findByEmail")

    Employee findByDepartmentIdNamed(@Param("email") int departmentId);

    // Derived query methods

    List<Employee> findByName(String name);

    List<Employee> findByDepartmentId(int DepartmentId);

    List<Employee> findByEmail(String email);

}

**DepartmentRepository.java🡪**

package com.code.employee.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import com.code.employee.entity.Department;

public interface DepartmentRepository  extends JpaRepository<Department, Integer>{

    // Derived query methods

    Department findByName(String name);

    //Custom JPQL query to find all department by name

    @Query("SELECT d FROM Department d WHERE d.name = :name")

    Department findByNameUsingJPQL(@Param("name") String name);

    //Custom JPQL query to find all departments with a specific name pattern

    // by default nativeQuery=false so when we want to use sql query, native query set to true

    @Query(value = "SELECT \* FROM departments d WHERE d.name LIKE %:pattern%", nativeQuery = true)

    List<Department> findByNamePattern(@Param("pattern") String pattern);

}

1. **Named Queries**:
   * Define and execute named queries with **@NamedQuery** and **@NamedQueries**.

**Employee.java🡪**

package com.code.employee.entity;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table

@Getter

@Setter

@NamedQueries({

    @NamedQuery(

        name = "Employee.findByEmail",

        query = "SELECT e FROM Employee e WHERE e.email = :email"

    ),

    @NamedQuery(

            name = "Employee.findByDepartmentId",

            query = "SELECT e FROM Employee e WHERE e.department = :department"

    )

})

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)

    private int id;

    @Column(name="name", nullable=false)

    private String name;

    @Column(name="email", nullable=false, unique=true)

    private String email;

    //relationship with department one employee can work in one department

    //under one department there will be many employees

    @ManyToOne

    //creating a foreign key department id ref to the pk of department

    @JoinColumn(name = "department\_id", nullable = false)

    private Department department;

    public Employee() {

        this.id = 0;

        this.name = null;

        this.email = null;

    }

}