**Exercise 3: Employee Management System - Creating Repositories**

**Business Scenario:**

Create repositories for Employee and Department entities to perform CRUD operations.

**Instructions:**

1. **Overview of Spring Data Repositories:**
   * Learn the benefits of using Spring Data repositories.

Spring Data repositories offer several key benefits:

1. **Simplified CRUD Operations**: Basic data access methods are provided out of the box, reducing boilerplate code.
2. **Consistent API**: A uniform approach to data access across various databases.
3. **Derived Query Methods**: Automatically generate query implementations based on method names.
4. **Pagination and Sorting**: Built-in support for handling large datasets efficiently.
5. **Integration with Spring**: Seamless use with Spring’s features like dependency injection and transaction management.

Overall, they streamline data access, enhance productivity, and reduce code complexity.

1. **Creating Repositories:**
   * Create **EmployeeRepository** and **DepartmentRepository** interfaces extending **JpaRepository**.
   * Define derived query methods in these repositories.

**DepartmentRepository.java 🡪**

package com.code.employee.repository;

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

import com.code.employee.entity.Department;

public interface DepartmentRepository  extends JpaRepository<Department, Integer>{

    Department findByName(String name);

}

**EmployeeRepository.java 🡪**

package com.code.employee.repository;

import java.util.List;

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

import com.code.employee.entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer>{

    List<Employee> findByName(String name);

    List<Employee> findByDepartmentId(int DepartmentId);

    List<Employee> findByEmail(String email);

}