Exercise 1: Employee Management System - Overview and Setup

Business Scenario:

You are developing an employee management system that will manage employee data, departments, and their relationships.

Instructions:

1. **Creating a Spring Boot Project:**
    - Initialize a new Spring Boot project named **EmployeeManagementSystem**.
    - Add dependencies: **Spring Data JPA, H2 Database, Spring Web, Lombok**.
2. **Configuring Application Properties:**
    - Configure **application.properties** for H2 database connection.

      spring.datasource.url=jdbc:h2:mem:testdb
      spring.datasource.driverClassName=org.h2.Driver
      spring.datasource.username=sa
      spring.datasource.password=password
      spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

Exercise 2: Employee Management System - Creating Entities

Business Scenario:

Define JPA entities for Employee and Department with appropriate relationships.

Instructions:

1. **Creating JPA Entities:**
    - Define **Employee** entity with fields: **id, name, email, department**.
    - Define **Department** entity with fields: **id, name**.
2. **Mapping Entities to Database Tables:**
    - Use annotations like **@Entity, @Table, @Id, @GeneratedValue**, etc.
    - Define one-to-many relationship between **Department** and **Employee**.

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

Exercise 4: Employee Management System - Implementing CRUD Operations

Business Scenario:

Implement CRUD operations for managing employees and departments.

Instructions:

1. **Basic CRUD Operations:**
    - Use **JpaRepository** methods to create, read, update, and delete employees and departments.
    - Implement RESTful endpoints for these operations using **EmployeeController** and **DepartmentController**.

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

Exercise 6: Employee Management System - Implementing Pagination and Sorting

Business Scenario:

Add pagination and sorting capabilities to your employee search functionality.

Instructions:

1. **Pagination:**
    - Implement pagination for the employee list using **Page** and **Pageable**.
2. **Sorting:**
    - Add sorting functionality to your queries.
    - Combine pagination and sorting in your search endpoint.

Exercise 7: Employee Management System - Enabling Entity Auditing

Business Scenario:

Implement auditing to track the creation and modification of employees and departments.

Instructions:

1. **Entity Auditing:**
    - Enable auditing in your application by configuring auditing properties.
    - Use annotations like **@CreatedBy, @LastModifiedBy, @CreatedDate**, and **@LastModifiedDate**.

Exercise 8: Employee Management System - Creating Projections

Business Scenario:

Create projections to fetch specific data subsets from the employee and department entities.

Instructions:

1. **Projections:**
    - Define interface-based and class-based projections.
    - Use **@Value** and constructor expressions to control the fetched data.

Exercise 9: Employee Management System - Customizing Data Source Configuration

Business Scenario:

Customize your data source configuration and manage multiple data sources.

Instructions:

1. **Spring Boot Auto-Configuration:**
    - Leverage Spring Boot auto-configuration for data sources.
2. **Externalizing Configuration:**
    - Externalize configuration with application.properties.
    - Manage multiple data sources within your application.

Exercise 10: Employee Management System - Hibernate-Specific Features

Business Scenario:

Leverage Hibernate-specific features to enhance your application's performance and capabilities.

Instructions:

1. **Hibernate-Specific Annotations:**
    - Use Hibernate-specific annotations to customize entity mappings.
2. **Configuring Hibernate Dialect and Properties:**
    - Configure Hibernate dialect and properties for optimal performance.
3. **Batch Processing:**
    - Implement batch processing with Hibernate for bulk operations.