

Hands on 1:

Spring Data JPA - Quick Example

Answer:

OrmLearnApplication.java:

```
package com.cognizant.ormlearn;
```

```
import java.util.List;
```

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
```

```
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.context.ApplicationContext;
```

```
import com.cognizant.ormlearn.model.Country;
```

```
import com.cognizant.ormlearn.service.CountryService;
```

```
@SpringBootApplication
```

```
public class OrmLearnApplication {
```

```
    private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);
```

```
    private static CountryService countryService;
```

```
    public static void main(String[] args) {
```

```
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
```

```
        countryService = context.getBean(CountryService.class);
```

```
        testGetAllCountries();
```

```
    }
```

```
    private static void testGetAllCountries() {
```

```
        LOGGER.info("Start");
```

```
        List<Country> countries = countryService.getAllCountries();  
        LOGGER.debug("countries={}", countries);  
        LOGGER.info("End");  
    }  
}
```

Country.java:

```
package com.cognizant.ormlearn.model;
```

```
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;
```

```
@Entity  
@Table(name = "country")  
public class Country {
```

```
    @Id  
    @Column(name = "co_code")  
    private String code;
```

```
    @Column(name = "co_name")  
    private String name;
```

```
    public String getCode() { return code; }  
    public void setCode(String code) { this.code = code; }
```

```
    public String getName() { return name; }  
    public void setName(String name) { this.name = name; }
```

```
@Override
```

```
public String toString() {  
    return "Country [code=" + code + ", name=" + name + "]";  
}  
}
```

CountryRepository.java:

```
package com.cognizant.ormlearn.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import com.cognizant.ormlearn.model.Country;
```

```
@Repository
```

```
public interface CountryRepository extends JpaRepository<Country, String> {  
}
```

CountryService.java:

```
package com.cognizant.ormlearn.service;
```

```
import java.util.List;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.repository.CountryRepository;
```

```
@Service
```

```
public class CountryService {
```

```
    @Autowired
```

```
    private CountryRepository countryRepository;
```

@Transactional

```
public List<Country> getAllCountries() {
    return countryRepository.findAll();
}
```

Output:

```
<terminated> [OrmLearnApplication (4) Java Application] C:\Users\Anuj\Downloads\OpenJDK7U-jdk_x64_windows_hotspot_17.0.10_7\jdk-17.0.10\bin\javaw.exe (06-Jul-2025, 0:08:00 pm - 8:08:15 pm elapsed: 0:00:14.429 [pid: 141100])
INFO org.hibernate.Version logVersion 4.1.10.Final
INFO .i.RegistrarFactoryInitiator initiateService 59 INFO000412: Hibernate ORM core version 5.4.10.Final
INFO SpringSessionUtilInfo addTransformer 87 No LoadTimeLeaving set; ignoring JPA class transformer
INFO .i.HikariDataSource checkFailFast 109 HikariPool-1 - Starting...
INFO c.z.h.HikariPool getConnections 122 HikariPool-1 - Start completed.
WARN o.horm.deprecation constructDialect 153 HHH00000025: MySQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the prop)
WARN o.horm.deprecation logSelectedDialect 101 HHH00000026: MySQLDialect has been deprecated; use org.hibernate.dialect.MySQLDialect instead
INFO o.h.o.connections.pooling logConnectionInfo 163 HHH10001005: Database info:
connecting through datasource 'HikariDataSource (HikariPool-1)'

ined/unknown
ined/unknown
efined/unknown
efined/unknown

DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(12, org.hibernate.type.descriptor.sql.internal.CapacityDependentDlType@e6b4ea6) replaced previous
DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.CapacityDependentDlType@6110856a) replaced prev:
DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.CapacityDependentDlType@5ee5bdac) replaced prev:
DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(003, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@464828d) replaced previous registr
DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@99de58e) replaced previous registr
DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(2084, org.hibernate.type.descriptor.sql.internal.CapacityDependentDlType@64f0be70) replaced prev:
DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(2085, org.hibernate.type.descriptor.sql.internal.CapacityDependentDlType@205ed5fe) replaced prev:
DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDependentDlType@699bdb90) replaced prev:
INFO p.i.JtaPlatformInitiator initiateService 59 HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integral
INFO rEntityManagerFactoryBean buildNativeEntityManagerFactory 447 Initial JPA EntityManagerFactory for persistence unit 'default'
INFO .OptionalLiveReloadServer startServer 59 LiveReload server is running on port 35729
INFO c.c.oOrmLearnApplication logStarted 59 Started OrmLearnApplication in 9.497 seconds (process running for 11.191)
INFO c.c.oOrmLearnApplication test GetAllCountries 27 Start
INFO org.hibernate.SQL logStatement 135 select c1_.co_code,c1_.co_name from country c1_
INFO c.c.oOrmLearnApplication test GetAllCountries 29 countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]
INFO c.c.oOrmLearnApplication test GetAllCountries 30 End
ShutdownHook INFO rEntityManagerFactoryBean destroy 660 Closing JPA EntityManagerFactory for persistence unit 'default'
ShutdownHook INFO c.z.h.HikariDataSource close 349 HikariPool-1 - Shutdown initiated...
ShutdownHook INFO c.z.h.HikariDataSource close 351 HikariPool-1 - Shutdown completed.
```

Difference between JPA, Hibernate and Spring Data JPA

Aspect	JPA (Java Persistence API)	Hibernate	Spring Data JPA
Type	Specification (JSR 338)	ORM Framework (JPA implementation)	Abstraction layer on top of JPA implementations
Purpose	Defines standard interfaces and rules for ORM	Implements JPA spec and adds extra features	Simplifies data access by reducing boilerplate
Implementation	No concrete implementation	Concrete implementation of JPA	Uses existing JPA providers (like Hibernate)
Transaction	No transaction management	Manages transactions manually	Manages transactions automatically
Boilerplate Code	N/A (only interfaces and specs)	Requires manual session and transaction code	Minimal code, CRUD methods auto-implemented
Example Use	Defines EntityManager, Entity, Query, etc.	Use SessionFactory, Session, Transaction	Use Repository interfaces with Spring framework

JPA:

A **standard API** (interface/specification) for managing persistence and ORM in Java.

Specifies how to map Java objects to database tables, query databases, and manage transactions.

No actual code implementation — just defines *what* to do, not *how*.

Multiple implementations exist (e.g., Hibernate, EclipseLink).

Example code:

```
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}
```

@Service

```
public class EmployeeService {  
    @Autowired  
    private EmployeeRepository employeeRepository;
```

@Transactional

```
public void addEmployee(Employee employee) {  
    employeeRepository.save(employee);  
}
```

Hibernate

An **ORM framework** that implements the JPA specification.

Provides concrete code for JPA interfaces plus **extra features** (e.g., caching, better performance tools).

Requires manual management of sessions and transactions unless you integrate with a container.

Example code:

```
public Integer addEmployee(Employee employee){  
    Session session = factory.openSession();  
    Transaction tx = null;  
    Integer employeeID = null;
```

```

try {
    tx = session.beginTransaction();
    employeeID = (Integer) session.save(employee);
    tx.commit();
} catch (HibernateException e) {
    if (tx != null) tx.rollback();
    e.printStackTrace();
} finally {
    session.close();
}
return employeeID;
}

```

Exercise 1: Configuring a Basic Spring Application

Scenario:

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

Steps:

1. Set Up a Spring Project:

- Create a Maven project named **LibraryManagement**.
- Add Spring Core dependencies in the **pom.xml** file.

2. Configure the Application Context:

- Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
- Define beans for **BookService** and **BookRepository** in the XML file.

3. Define Service and Repository Classes:

- Create a package **com.library.service** and add a class **BookService**.
- Create a package **com.library.repository** and add a class **BookRepository**.

4. Run the Application:

- o Create a main class to load the Spring context and test the configuration.

Answer:

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.library</groupId>
  <artifactId>LibraryManagement</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
    <!-- Spring Context for ApplicationContext and Beans -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.32</version>
    </dependency>
  </dependencies>

</project>
```

BookRepository.java

```
package com.library.repository;

public class BookRepository {

  public void saveBook(String bookName) {
    System.out.println("Book saved: " + bookName);
  }
}
```

```
    }  
}
```

BookService.java

```
package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
    private BookRepository bookRepository;  
  
    public void setBookRepository(BookRepository bookRepository) {  
        this.bookRepository = bookRepository;  
    }  
  
    public void addBook(String bookName) {  
        System.out.println("Adding book: " + bookName);  
        bookRepository.saveBook(bookName);  
    }  
}
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
       xsi:schemaLocation="  
           http://www.springframework.org/schema/beans  
           http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
    <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
    <bean id="bookService" class="com.library.service.BookService">
```

```
<property name="bookRepository" ref="bookRepository" />  
</bean>  
  
</beans>  
MainApp.java  
package library;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import com.library.service.BookService;  
  
public class MainApp {  
    public static void main(String[] args) {  
        ApplicationContext context =  
            new ClassPathXmlApplicationContext("applicationContext.xml");  
  
        BookService service = (BookService) context.getBean("bookService");  
        service.addBook("Spring in Action");  
    }  
}
```

Output:

```
1 package library;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import com.library.service.BookService;
6
7 public class MainApp {
8     @Bean
9     public static void main(String[] args) {
10         ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
11
12         BookService service = context.getBean(BookService.class);
13         service.addBook("Spring in Action");
14     }
15 }
16
```

<terminated> MainApp [Java Application] C:\Users\anju.p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.fine

Adding book: Spring in Action
Book saved: Spring in Action

Exercise 2: Implementing Dependency Injection

Scenario:

In the library management application, you need to manage the dependencies between the `BookService` and `BookRepository` classes using Spring's IoC and DI.

Steps:

1. **Modify the XML Configuration:**
 - Update `applicationContext.xml` to wire `BookRepository` into `BookService`.
2. **Update the BookService Class:**
 - Ensure that `BookService` class has a setter method for `BookRepository`.
3. **Test the Configuration:**
 - Run the `LibraryManagementApplication` main class to verify the dependency injection.

Answer:

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
```

```
https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>
<artifactId>LibraryManagementApp</artifactId>
<version>1.0</version>

<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.34</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.3.34</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>5.3.34</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jcl</artifactId>
        <version>5.3.34</version>
    </dependency>
</dependencies>

<build>
    <sourceDirectory>src/main/java</sourceDirectory>
```

```
<resources>
  <resource>
    <directory>src/main/resources</directory>
  </resource>
</resources>
<plugins>
  <plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.10.1</version>
    <configuration>
      <release>17</release>
    </configuration>
  </plugin>
</plugins>
</build>
</project>
```

BookRepository.java

```
package com.example.repository;

public class BookRepository {
  public void save(String book) {
    System.out.println("Saving book: " + book);
  }
}
```

BookService.java

```
package com.example.service;

import com.example.repository.BookRepository;

public class BookService {
```

```
private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {
    this.bookRepository = bookRepository;
}

public void addBook(String book) {
    System.out.println("Adding book: " + book);
    bookRepository.save(book);
}
```

Library.java

```
package com.example;

import com.example.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Library {
    public static void main(String[] args) {
        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService service = context.getBean("bookService", BookService.class);
        service.addBook("Java Programming");
    }
}
```

Output:

```
1 package com.example;
2
3 import com.example.service.BookService;
4 import org.springframework.context.ApplicationContext;
5 import org.springframework.context.support.ClassPathXmlApplicationContext;
6
7 public class Library {
8     public static void main(String[] args) {
9         ApplicationContext context =
10             new ClassPathXmlApplicationContext("applicationContext.xml");
11
12         BookService service = context.getBean("bookService", BookService.class);
13         service.addBook("Java Programming");
14     }
15 }
16
```

Exercise 4: Creating and Configuring a Maven Project

Scenario:

You need to set up a new Maven project for the library management application and add Spring dependencies.

Steps:

1. **Create a New Maven Project:**
 - o Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
 - o Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
 - o Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

Answer:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>LibraryManagement</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.36</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aop</artifactId>
      <version>5.3.36</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.3.36</version>
    </dependency>
  </dependencies>

  <build>
```

```

<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.11.0</version>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
    </configuration>
  </plugin>
</plugins>
</build>

</project>

```

Exercise 5: Configuring the Spring IoC Container

Scenario:

The library management application requires a central configuration for beans and dependencies.

Steps:

1. Create Spring Configuration File:

- Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
- Define beans for **BookService** and **BookRepository** in the XML file.

2. Update the BookService Class:

- Ensure that the **BookService** class has a setter method for **BookRepository**.

3. Run the Application:

Create a main class to load the Spring context and test the configuration

Answer:

Pom.xml

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

```

<https://maven.apache.org/xsd/maven-4.0.0.xsd>">

```
<modelVersion>4.0.0</modelVersion>
```

```
<groupId>com.example</groupId>
<artifactId>LibraryApp</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

```
<dependencies>
```

```
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.30</version>
    </dependency>
```

```
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.36</version>
    </dependency>
```

```
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-simple</artifactId>
        <version>1.7.36</version>
    </dependency>
```

```
</dependencies>
```

```
<build>
```

```
    <plugins>
```

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.11.0</version>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
</plugin>
</plugins>
</build>
```

```
</project>
BookService.java
package com.example.service;
```

```
import com.example.repository.BookRepository;

public class BookService {
```

```
  private BookRepository bookRepository;
```

```
  public void setBookRepository(BookRepository bookRepository) {
    this.bookRepository = bookRepository;
  }
```

```
  public void addBook(String title) {
    System.out.println("Adding book: " + title);
    bookRepository.saveBook(title);
  }
}
```

```
BookRepository.java
```

```
package com.example.repository;

public class BookRepository {

    public void saveBook(String title) {
        System.out.println("Book " + title + " saved to the repository.");
    }
}
```

LibraryApp.java

```
package com.example;

import com.example.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");
        bookService.addBook("Spring Framework Basics");
    }
}
```

Output:

The screenshot shows an IDE interface with the following components:

- Left Panel (Code Editor):** Displays the `LibraryApp.java` file. The code imports `com.example.service.BookService`, `org.springframework.context.ApplicationContext`, and `org.springframework.context.support.ClassPathXmlApplicationContext`. It defines a `main` method that creates a `ClassPathXmlApplicationContext` and gets a `BookService` bean from it, adding a book titled "Spring Framework Basics".
- Right Panels:**
 - Task List:** Shows a single task entry.
 - Outline:** Shows the package structure: `com.example` and the class `LibraryApp`.
- Bottom Panel (Console):** Shows the terminal output of the application's execution. It includes the command run, the application's name, the Java version used, and the log message indicating a book was added and saved to the repository.

Exercise 7: Implementing Constructor and Setter Injection

Scenario:

The library management application requires both constructor and setter injection for better control over bean initialization.

Steps:

1. Configure Constructor Injection:
 - o Update `applicationContext.xml` to configure constructor injection for `BookService`.
2. Configure Setter Injection:
 - o Ensure that the `BookService` class has a setter method for `BookRepository` and configure it in `applicationContext.xml`.
3. Test the Injection:
 - o Run the `LibraryManagementApplication` main class to verify both constructor and setter injection.

Answer

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>Library</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.34</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>5.3.34</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
```

```
<artifactId>maven-compiler-plugin</artifactId>
<version>3.10.1</version>
<configuration>
  <source>1.8</source>
  <target>1.8</target>
</configuration>
</plugin>
</plugins>
</build>
```

```
</project>
```

BookService.java

```
package com.example;

public class BookService {
    private BookRepository bookRepository;

    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
        System.out.println("Constructor Injection used.");
    }

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
        System.out.println("Setter Injection used.");
    }

    public void showBooks() {
        bookRepository.displayBooks();
    }
}
```

BookRepository.java

```
package com.example;

public class BookRepository {
    public void displayBooks() {
        System.out.println("Displaying books from the repository...");
    }
}
```

LibraryManagementApplication.java

```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");
        bookService.showBooks();
    }
}
```

Output:

```
1 package com.example;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class LibraryManagementApplication {
7     public static void main(String[] args) {
8         ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
9
10        BookService bookService = (BookService) context.getBean("bookService");
11        bookService.showBooks();
12    }
13 }
14
```

Problems @ Javadoc Declaration Console X
<terminated> LibraryManagementApplication [Java Application] C:\Users\anju\Downloads\OpenJDK17U-jdk_x64_windows_hotspot_17.0.10_7\jdk-17.0.10+7b
Constructor Injection used.
Setter Injection used.
Displaying books from the repository...

Exercise 9: Creating a Spring Boot Application

Scenario:

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

Steps:

1. **Create a Spring Boot Project:**
 - o Use **Spring Initializr** to create a new Spring Boot project named **LibraryManagement**.
2. **Add Dependencies:**
 - o Include dependencies for **Spring Web**, **Spring Data JPA**, and **H2 Database**.
3. **Create Application Properties:**
 - o Configure database connection properties in **application.properties**.
4. **Define Entities and Repositories:**
 - o Create **Book** entity and **BookRepository** interface.
5. **Create a REST Controller:**
 - o Create a **BookController** class to handle CRUD operations.
6. **Run the Application:**
 - o Run the Spring Boot application and test the REST endpoints.

Answer:

Book.java

```
package com.example;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
public class Book {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    private String title;
```

```
    private String author;
```

```
    private String isbn;
```

```
// Getters and Setters
```

```
    public Long getId() { return id; }
```

```
    public void setId(Long id) { this.id = id; }
```

```
    public String getTitle() { return title; }
```

```
    public void setTitle(String title) { this.title = title; }
```

```
    public String getAuthor() { return author; }
```

```
    public void setAuthor(String author) { this.author = author; }
```

```
    public String getIsbn() { return isbn; }
```

```
    public void setIsbn(String isbn) { this.isbn = isbn; }
```

```
}
```

BookController.java

```
package com.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookRepository bookRepository;

    @PostMapping
    public Book addBook(@RequestBody Book book) {
        return bookRepository.save(book);
    }

    @GetMapping
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    @GetMapping("/{id}")
    public Optional<Book> getBookById(@PathVariable Long id) {
        return bookRepository.findById(id);
    }

    @PutMapping("/{id}")
```

```
public Book updateBook(@PathVariable Long id, @RequestBody Book bookDetails) {  
    Book book = bookRepository.findById(id).orElseThrow();  
    book.setTitle(bookDetails.getTitle());  
    book.setAuthor(bookDetails.getAuthor());  
    book.setIsbn(bookDetails.getIsbn());  
    return bookRepository.save(book);  
}  
  
@DeleteMapping("/{id}")
```

```
public void deleteBook(@PathVariable Long id) {  
    bookRepository.deleteById(id);  
}  
}
```

BookRepository.java

```
package com.example;
```

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

```
public interface BookRepository extends JpaRepository<Book, Long> {  
}
```

LibraryManagementApplication.java

```
package com.example;
```

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

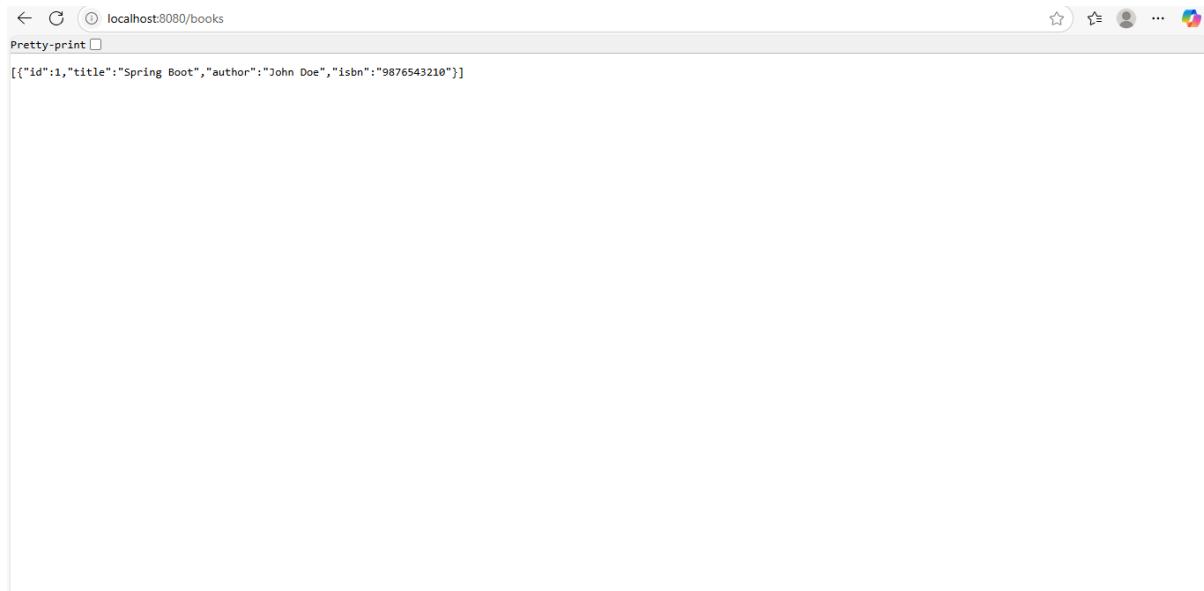
```
@SpringBootApplication
```

```
public class LibraryManagementApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(LibraryManagementApplication.class, args);  
    }  
}
```

Output:



Exercise 10: Implement services for managing Country

CountryServiceApplication.java

```
package com.example.country_service;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class CountryServiceApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(CountryServiceApplication.class, args);  
    }  
}
```

```
}
```

```
Country.java
```

```
import jakarta.persistence.Table;
```

```
@Entity
```

```
@Table(name = "country")
```

```
public class Country {
```

```
    @Id
```

```
    @Column(name = "co_code")
```

```
    private String coCode;
```

```
    @Column(name = "co_name")
```

```
    private String coName;
```

```
    public String getCoCode() {
```

```
        return coCode;
```

```
    }
```

```
    public void setCoCode(String coCode) {
```

```
        this.coCode = coCode;
```

```
    }
```

```
    public String getCoName() {
```

```
        return coName;
```

```
    }
```

```
    public void setCoName(String coName) {
```

```
        this.coName = coName;
```

```
    }
```

```
}
```

CountryController.java

```
package com.example.countryservice;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/countries")
public class CountryController {

    @Autowired
    private CountryService countryService;

    @GetMapping("/{code}")
    public Optional<Country> getCountryByCode(@PathVariable String code) {
        return countryService.findCountryByCode(code);
    }

    @PostMapping
    public Country addCountry(@RequestBody Country country) {
        return countryService.addCountry(country);
    }

    @PutMapping
    public Country updateCountry(@RequestBody Country country) {
        return countryService.updateCountry(country);
    }
}
```

```
}
```



```
@DeleteMapping("/{code}")  
public void deleteCountry(@PathVariable String code) {  
    countryService.deleteCountry(code);  
}
```



```
@GetMapping("/search")  
public List<Country> searchCountries(@RequestParam String name) {  
    return countryService.findCountriesByPartialName(name);  
}  
}
```

CountryRepository.java

```
package com.example.countryservice;
```

```
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.List;
```

```
public interface CountryRepository extends JpaRepository<Country, String> {  
    List<Country> findByCoNameContainingIgnoreCase(String name);  
}
```

CountryService.java

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import java.util.List;  
import java.util.Optional;
```

@Service

```
public class CountryService {
```

```
    @Autowired
```

```
private CountryRepository countryRepository;

public Optional<Country> findCountryByCode(String code) {
    return countryRepository.findById(code);
}

public Country addCountry(Country country) {
    return countryRepository.save(country);
}

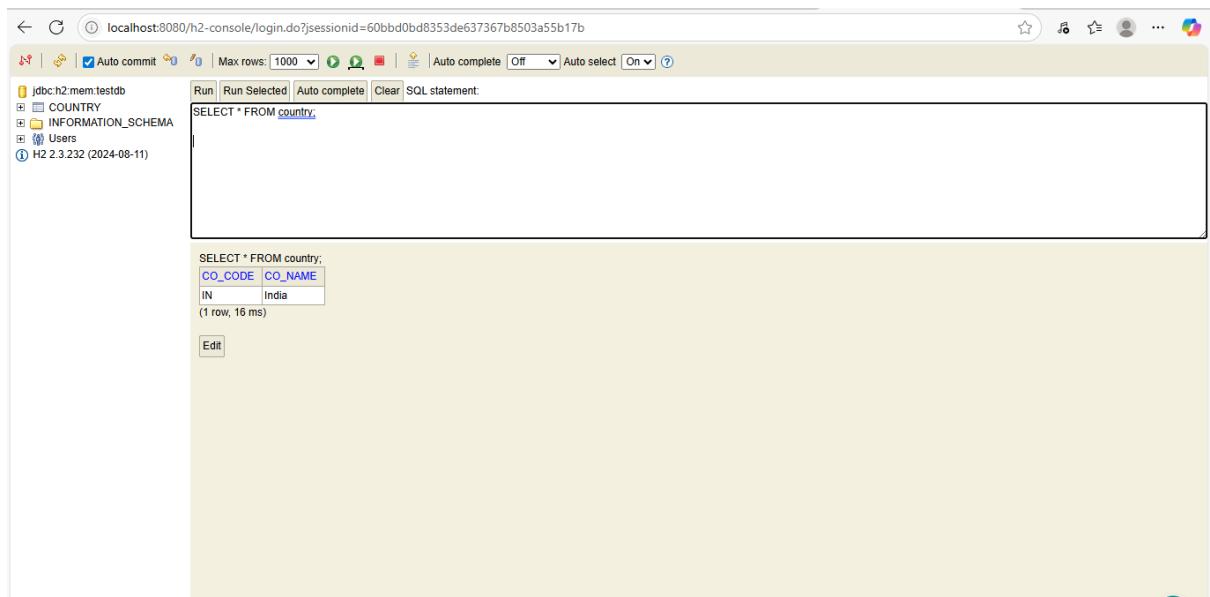
public Country updateCountry(Country country) {
    return countryRepository.save(country);
}

public void deleteCountry(String code) {
    countryRepository.deleteById(code);
}

public List<Country> findCountriesByPartialName(String partialName) {
    return countryRepository.findByNameContainingIgnoreCase(partialName);
}

data.sql
INSERT INTO country (co_code, co_name) VALUES ('IN', 'India');
```

Output:



Find a country based on country code

OrmLearnApplication.java

```
package com.cognizant.spring_learn;
```

```
import com.cognizant.spring_learn.model.Country;
import com.cognizant.spring_learn.service.CountryService;
import com.cognizant.spring_learn.service.exception.CountryNotFoundException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
```

```
@SpringBootApplication
```

```
public class OrmLearnApplication {
```

```
    private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);
```

```

public static void main(String[] args) {
    ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
    CountryService countryService = context.getBean(CountryService.class);
    testGetCountryByCode(countryService);
}

private static void testGetCountryByCode(CountryService countryService) {
    LOGGER.info("Start - testGetCountryByCode");
    try {
        Country country = countryService.findCountryByCode("IN");

        LOGGER.info("Country: {}", country);
    } catch (CountryNotFoundException e) {
        LOGGER.error("Country not found: {}", e.getMessage());
    }
    LOGGER.info("End - testGetCountryByCode");
}

```

Country.java

```
package com.cognizant.spring_learn.model;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
@Table(name = "country")
```

```
public class Country {
```

```
    @Id
```

```
    @Column(name = "co_code")
```

```
private String code;

@Column(name = "co_name")
private String name;

// Getters & Setters
public String getCode() {
    return code;
}

public void setCode(String code) {
    this.code = code;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}
```

```
@Override
public String toString() {
    return "Country [code=" + code + ", name=" + name + "]";
}
```

CountryNotFoundException.java

```
package com.cognizant.spring_learn.service.exception;
```

```
public class CountryNotFoundException extends Exception {
```

```
public CountryNotFoundException() {
    super();
}
```

```
}

public CountryNotFoundException(String message) {
    super(message);
}

}
```

CountryRepository.java

```
package com.cognizant.spring_learn.repository;
```

```
import com.cognizant.spring_learn.model.Country;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
public interface CountryRepository extends JpaRepository<Country, String> {
```

```
}
```

OrmLearnApplicationTests.java

```
package com.cognizant.spring_learn;
```

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
```

```
@SpringBootTest
```

```
class OrmLearnApplicationTests {
```

```
    @Test
```

```
    void contextLoads() {
    }
```

```
}
```

Output:

The screenshot shows two windows. The top window is an IDE (Eclipse) displaying application logs for 'OrmLearnApplication [Java Application]'. The logs show the application starting up, connecting to a HikariDataSource, and initializing a Spring Data JPA repository. The bottom window is a web browser showing the H2 console at 'localhost:8080/h2-console/login.do?sessionid=82aed558d19a2235d35b0429d590343e'. It displays a table named 'COUNTRY' with two rows: 'IN' (India) and 'US' (United States).

```

2025-07-06T01:17:56.895+05:30 INFO 129616 --- [main] c.c.spring.learnOrmLearnApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-06T01:17:58.820+05:30 INFO 129616 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-06T01:17:58.925+05:30 INFO 129616 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 87 ms. Found 1 JPA repository interface.
2025-07-06T01:18:00.150+05:30 INFO 129616 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-07-06T01:18:00.284+05:30 INFO 129616 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-06T01:18:00.295+05:30 INFO 129616 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-06T01:18:00.372+05:30 INFO 129616 --- [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2025-07-06T01:18:00.649+05:30 INFO 129616 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3284 ms
2025-07-06T01:18:01.278+05:30 INFO 129616 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-06T01:18:01.285+05:30 INFO 129616 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA
2025-07-06T01:18:01.629+05:30 INFO 129616 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-06T01:18:01.789+05:30 INFO 129616 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-06T01:18:01.904+05:30 INFO 129616 --- [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2025-07-06T01:18:01.606+05:30 INFO 129616 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-06T01:18:01.606+05:30 INFO 129616 --- [main] org.hibernate.orm.deprecation : HHH0000025: H2Dialect does not need to be specified explicitly using 'hibernate.dialect'
2025-07-06T01:18:02.814+05:30 INFO 129616 --- [main] org.hibernate.orm.connections.Pooling : HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 2.3.232
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
Hibernate: drop table if exists country cascade
Hibernate: create table country (co_code varchar(255) not null, co_name varchar(255), primary key (co_code))
2025-07-06T01:18:04.663+05:30 INFO 129616 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-06T01:18:05.195+05:30 INFO 129616 --- [main] jpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be executed via View Objects, even if they are not explicitly marked as @Query
2025-07-06T01:18:05.702+05:30 INFO 129616 --- [main] o.s.b.a.h.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:test'
2025-07-06T01:18:05.913+05:30 INFO 129616 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-06T01:18:05.939+05:30 INFO 129616 --- [main] c.c.spring.learnOrmLearnApplication : Started OrmLearnApplication in 10.949 seconds (process running for 12.188)
2025-07-06T01:18:05.947+05:30 INFO 129616 --- [main] c.c.spring.learnOrmLearnApplication : Start - testGetCountryByCode
Hibernate: select cl_0.co_code, cl_0.co_name from country cl_0 where cl_0.co_code=?
2025-07-06T01:18:06.218+05:30 ERROR 129616 --- [main] c.c.spring.learnOrmLearnApplication : Country not found: Country not found with code: IN
2025-07-06T01:18:06.219+05:30 INFO 129616 --- [main] c.c.spring.learnOrmLearnApplication : End - testGetCountryByCode

```

Excercise: Add a new Country:

Answer:

```
package com.cognizant.spring_learn.service;
```

```
import com.cognizant.spring_learn.model.Country;
import com.cognizant.spring_learn.repository.CountryRepository;
import com.cognizant.spring_learn.service.exception.CountryNotFoundException;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Optional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public Country findCountryByCode(String code) throws CountryNotFoundException {
        Optional<Country> result = countryRepository.findById(code);
        if (result.isPresent()) {
            return result.get();
        } else {
            throw new CountryNotFoundException("Country not found with code: " + code);
        }
    }

    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
}

package com.cognizant.spring_learn.repository;

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

```
import com.cognizant.spring_learn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {
}

package com.cognizant.spring_learn.model;

import jakarta.persistence.*;

@Entity
@Table(name = "country")
public class Country {

    @Id
    private String code;
    private String name;

    // Getters and Setters

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
// toString  
  
@Override  
  
public String toString() {  
    return "Country [code=" + code + ", name=" + name + "]";  
}  
  
}  
  
package com.cognizant.spring_learn.service.exception;  
  
public class CountryNotFoundException extends Exception {  
    public CountryNotFoundException(String message) {  
        super(message);  
    }  
}  
  
import org.springframework.context.ApplicationContext;  
  
@SpringBootApplication  
public class OrmLearnApplication {  
  
    public static void main(String[] args) {  
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);  
        testAddCountry(context);  
    }  
  
    public static void testAddCountry(ApplicationContext context) {  
        CountryService countryService = context.getBean(CountryService.class);  
  
        Country newCountry = new Country();  
        newCountry.setCode("JP");  
        newCountry.setName("Japan");  
    }  
}
```

```
countryService.addCountry(newCountry);

try {
    Country country = countryService.findCountryByCode("JP");
    System.out.println("Added Country: " + country);
} catch (CountryNotFoundException e) {
    System.out.println(e.getMessage());
}

}
```

Output:

```
PS C:\Users\anju\eclipse-workspace\country-service> mvn spring-boot:run -X-----
```

Exercise:Write queries on country table using Query Methods

Answer:

```
package com.example.ormlearn.model;
```

```
import jakarta.persistence.*;
```

@Entity

```
@Table(name = "country")
```

```
public class Country {
```

```
@Id  
private String code;  
  
private String name;  
  
public String getCode() {  
    return code;  
}  
  
public void setCode(String code) {  
    this.code = code;  
}  
  
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
@Override  
public String toString() {  
    return code + " " + name;  
}  
}  
  
package com.example.ormlearn;  
  
import com.example.ormlearn.model.Country;  
import com.example.ormlearn.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private CountryRepository countryRepository;

    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        System.out.println("Countries containing 'ou':");
        List<Country> countries = countryRepository.findByNameContainingIgnoreCase("ou");
        countries.forEach(System.out::println);

        System.out.println("\nCountries containing 'ou' (Sorted Ascending):");
        List<Country> sortedCountries =
        countryRepository.findByNameContainingIgnoreCaseOrderByAsc("ou");
        sortedCountries.forEach(System.out::println);

        System.out.println("\nCountries starting with 'Z':");
        List<Country> zCountries = countryRepository.findByNameStartingWithIgnoreCase("Z");
        zCountries.forEach(System.out::println);
    }
}
```

```
}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Country;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CountryRepository extends JpaRepository<Country, String> {

    List<Country> findByNameContainingIgnoreCase(String name);

    List<Country> findByNameContainingIgnoreCaseOrderByAsc(String name);

    List<Country> findByNameStartingWithIgnoreCase(String prefix);

}

INSERT INTO country (code, name) VALUES
('BV', 'Bouvet Island'),
('DJ', 'Djibouti'),
('GP', 'Guadeloupe'),
('GS', 'South Georgia and the South Sandwich Islands'),
('LU', 'Luxembourg'),
('SS', 'South Sudan'),
('TF', 'French Southern Territories'),
('UM', 'United States Minor Outlying Islands'),
('ZA', 'South Africa'),
('ZM', 'Zambia'),
('ZW', 'Zimbabwe');
```

Output:

```
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Declaration Console Properties
OrmLearnApplication [1 Java Application] C:\Users\enju\Downloads\OpenJDK17U-jdk_x64_windows_hotspot_17.0.10_7\bin\javaw.exe (06-Jul-2025, 9:48:39 am elapsed: 0:01:27) [pid: 11104]
2025-07-06T09:48:45.242+05:30 INFO 111044 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform'
Hibernate: drop table if exists country cascade
Hibernate: create table country (code varchar(255) not null, name varchar(255), primary key (code))
2025-07-06T09:48:45.275+05:30 INFO 111044 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-06T09:48:45.728+05:30 WARN 111044 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be executed during a JSP rendering. Consider to disable it in a production environment.
2025-07-06T09:48:46.088+05:30 INFO 111044 --- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:test'
2025-07-06T09:48:46.318+05:30 INFO 111044 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-06T09:48:46.327+05:30 INFO 111044 --- [main] c.example.ormlearnOrmLearnApplication : Started OrmLearnApplication in 5.6 seconds (process running for 6.033)
Countries containing 'ou':
Hibernate: select c1_0.code,c1_0.name from country c1_0 where upper(c1_0.name) like upper(?) escape '\'
BV Bouvet Island
DJ Djibouti
GP Guadeloupe
GS South Georgia and the South Sandwich Islands
LU Luxembourg
SS South Sudan
TF French Southern Territories
UM United States Minor Outlying Islands
ZA South Africa

Countries containing 'ou' (Sorted Ascending):
Hibernate: select c1_0.code,c1_0.name from country c1_0 where upper(c1_0.name) like upper(?) escape '\' order by c1_0.name
BV Bouvet Island
DJ Djibouti
TF French Southern Territories
GP Guadeloupe
LU Luxembourg
ZA South Africa
GS South Georgia and the South Sandwich Islands
SS South Sudan
UM United States Minor Outlying Islands

Countries starting with 'Z':
Hibernate: select c1_0.code,c1_0.name from country c1_0 where upper(c1_0.name) like upper(?) escape '\'
ZM Zambia
ZW Zimbabwe
```

Exercise: Write queries on stock table using Query Methods

Answer:

```
package com.example.ormlearn;

import com.example.ormlearn.model.Stock;
import com.example.ormlearn.repository.StockRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.math.BigDecimal;
import java.time.LocalDate;
import java.util.List;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {
```

@Autowired

```
private StockRepository stockRepository;

public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
}

@Override
public void run(String... args) {
    System.out.println("Query 1 - Facebook stocks in Sep 2019:");
    List<Stock> fbStocks = stockRepository.findByCodeAndDateBetween(
        "FB", LocalDate.of(2019, 9, 1), LocalDate.of(2019, 9, 30));
    fbStocks.forEach(System.out::println);

    System.out.println("\nQuery 2 - Google stocks with price > 1250:");
    List<Stock> googlStocks = stockRepository.findByCodeAndCloseGreaterThanOrEqual(
        "GOOGL", new BigDecimal("1250"));
    googlStocks.forEach(System.out::println);

    System.out.println("\nQuery 3 - Top 3 Highest Volume:");
    List<Stock> topVolumeStocks = stockRepository.findTop3ByOrderByVolumeDesc();
    topVolumeStocks.forEach(System.out::println);

    System.out.println("\nQuery 4 - Lowest 3 Netflix closing prices:");
    List<Stock> lowNetflixStocks = stockRepository.findTop3ByCodeOrderByCloseAsc("NFLX");
    lowNetflixStocks.forEach(System.out::println);
}
```

```
package com.example.ormlearn.model;

import javax.persistence.*;
import java.math.BigDecimal;
import java.time.LocalDate;

@Entity
@Table(name = "stock")

public class Stock {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "st_id")
    private Integer id;

    @Column(name = "st_code")
    private String code;

    @Column(name = "st_date")
    private LocalDate date;

    @Column(name = "st_open")
    private BigDecimal open;

    @Column(name = "st_close")
    private BigDecimal close;

    @Column(name = "st_volume")
    private BigDecimal volume;

    // Getters and Setters
}
```

```
public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getCode() {
    return code;
}

public void setCode(String code) {
    this.code = code;
}

public LocalDate getDate() {
    return date;
}

public void setDate(LocalDate date) {
    this.date = date;
}

public BigDecimal getOpen() {
    return open;
}

public void setOpen(BigDecimal open) {
    this.open = open;
}
```

```
public BigDecimal getClose() {
    return close;
}

public void setClose(BigDecimal close) {
    this.close = close;
}

public BigDecimal getVolume() {
    return volume;
}

public void setVolume(BigDecimal volume) {
    this.volume = volume;
}

@Override
public String toString() {
    return "Stock{" +
        "code='" + code + '\'' +
        ", date=" + date +
        ", open=" + open +
        ", close=" + close +
        ", volume=" + volume +
        '}';
}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Stock;
```

```

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

import java.math.BigDecimal;
import java.time.LocalDate;
import java.util.List;

public interface StockRepository extends JpaRepository<Stock, Integer> {

    List<Stock> findByCodeAndDateBetween(String code, LocalDate startDate, LocalDate endDate);

    List<Stock> findByCodeAndCloseGreaterThanOrEqual(String code, BigDecimal price);

    List<Stock> findTop3ByOrderByVolumeDesc();

    List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

```

Output:

```

mysql> SELECT * FROM country LIMIT 5;
ERROR 1146 (42S02): Table 'ormLearn.country' doesn't exist
mysql> SELECT * FROM stock LIMIT 5;
+-----+-----+-----+-----+-----+-----+
| st_id | st_code | st_date | st_open | st_close | st_volume |
+-----+-----+-----+-----+-----+-----+
| 1 | FB | 2019-09-03 | 184.00 | 182.39 | 9779400.00 |
| 2 | FB | 2019-09-04 | 184.65 | 187.14 | 11308000.00 |
| 3 | FB | 2019-09-05 | 188.53 | 190.90 | 13876700.00 |
| 4 | GOOGL | 2019-04-22 | 1236.67 | 1253.76 | 954200.00 |
| 5 | GOOGL | 2019-04-23 | 1256.64 | 1270.59 | 1593400.00 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.043 sec)

mysql> SELECT * FROM stock WHERE st_code = 'FB' ORDER BY st_date DESC LIMIT 5;
+-----+-----+-----+-----+-----+-----+
| st_id | st_code | st_date | st_open | st_close | st_volume |
+-----+-----+-----+-----+-----+-----+
| 3 | FB | 2019-09-05 | 188.53 | 190.90 | 13876700.00 |
| 9 | FB | 2019-09-05 | 188.53 | 190.90 | 13876700.00 |
| 2 | FB | 2019-09-04 | 184.65 | 187.14 | 11308000.00 |
| 8 | FB | 2019-09-04 | 184.65 | 187.14 | 11308000.00 |
| 1 | FB | 2019-09-03 | 184.00 | 182.39 | 9779400.00 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.059 sec)

mysql> SELECT st_code, AVG(st_close) as avg_close FROM stock GROUP BY st_code;
+-----+-----+
| st_code | avg_close |
+-----+-----+
| FB | 186.810000 |
| GOOGL | 1262.175000 |
| NFLX | 233.880000 |
+-----+-----+

```

```
MySQL 9.3 Command Line Cli X + ○
+-----+-----+-----+-----+-----+-----+
| st_id | st_code | st_date | st_open | st_close | st_volume |
+-----+-----+-----+-----+-----+-----+
| 3 | FB | 2019-09-05 | 188.53 | 190.90 | 13876700.00 |
| 9 | FB | 2019-09-05 | 188.53 | 190.90 | 13876700.00 |
| 2 | FB | 2019-09-04 | 184.65 | 187.14 | 11308000.00 |
| 8 | FB | 2019-09-04 | 184.65 | 187.14 | 11308000.00 |
| 1 | FB | 2019-09-03 | 184.00 | 182.39 | 9779400.00 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.059 sec)

mysql> SELECT st_code, AVG(st_close) as avg_close FROM stock GROUP BY st_code;
+-----+-----+
| st_code | avg_close |
+-----+-----+
| FB | 186.810000 |
| GOOGL | 1262.175000 |
| NFLX | 233.880000 |
+-----+-----+
3 rows in set (0.109 sec)

mysql> SELECT COUNT(*) FROM stock;
+-----+
| COUNT(*) |
+-----+
| 12 |
+-----+
1 row in set (0.185 sec)

mysql> |
```

Create payroll tables and bean mapping

Answer:

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.model.Employee;
import com.cognizant.orm_learn.repository.EmployeeRepository;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;

import java.util.List;

@SpringBootApplication
public class OrmLearnApplication {

    public static void main(String[] args) {
```

```

ConfigurableApplicationContext context = SpringApplication.run(OrmLearnApplication.class,
args);

EmployeeRepository employeeRepository = context.getBean(EmployeeRepository.class);

System.out.println("Fetching Employee Details...");
List<Employee> employees = employeeRepository.findAll();

if (employees.isEmpty()) {
    System.out.println("No Employee records found.");
} else {
    System.out.println("Employee records:");
    for (Employee employee : employees) {
        System.out.println(employee);
    }
}

context.close();
}

}

package com.cognizant.orm_learn.model;

import jakarta.persistence.*;
import java.util.Set;

@Entity
@Table(name = "department")
public class Department {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
}

```

```
@Column(name = "name")
private String name;

@OneToMany(mappedBy = "department")
private Set<Employee> employees;
}

package com.cognizant.orm_learn.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Skill {

    @Id
    private int id;

    private String name;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
}
```

```
}
```

```
public void setName(String name) {  
    this.name = name;  
}  
}  
  
package com.cognizant.orm_learn.repository;  
  
import com.cognizant.orm_learn.model.Department;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface DepartmentRepository extends JpaRepository<Department, Integer> {  
}  
  
package com.cognizant.orm_learn.repository;  
  
import com.cognizant.orm_learn.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}  
  
package com.cognizant.orm_learn.repository;  
  
import com.cognizant.orm_learn.model.Skill;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository
```

```
public interface SkillRepository extends JpaRepository<Skill, Integer> {  
}
```

```
CREATE TABLE IF NOT EXISTS department (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS skill (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS employee (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(100),  
    salary DOUBLE,  
    permanent BOOLEAN,  
    date_of_birth DATE,  
    department_id INT,  
    FOREIGN KEY (department_id) REFERENCES department(id)  
);
```

```
CREATE TABLE IF NOT EXISTS employee_skill (  
    employee_id INT,  
    skill_id INT,  
    PRIMARY KEY (employee_id, skill_id),  
    FOREIGN KEY (employee_id) REFERENCES employee(id),  
    FOREIGN KEY (skill_id) REFERENCES skill(id)  
);
```

```
INSERT INTO department (name) VALUES ('HR'), ('Finance'), ('Engineering');
```

```
INSERT INTO skill (name) VALUES ('Java'), ('Python'), ('SQL'), ('Communication');
```

```
INSERT INTO employee (name, salary, permanent, date_of_birth, department_id)
```

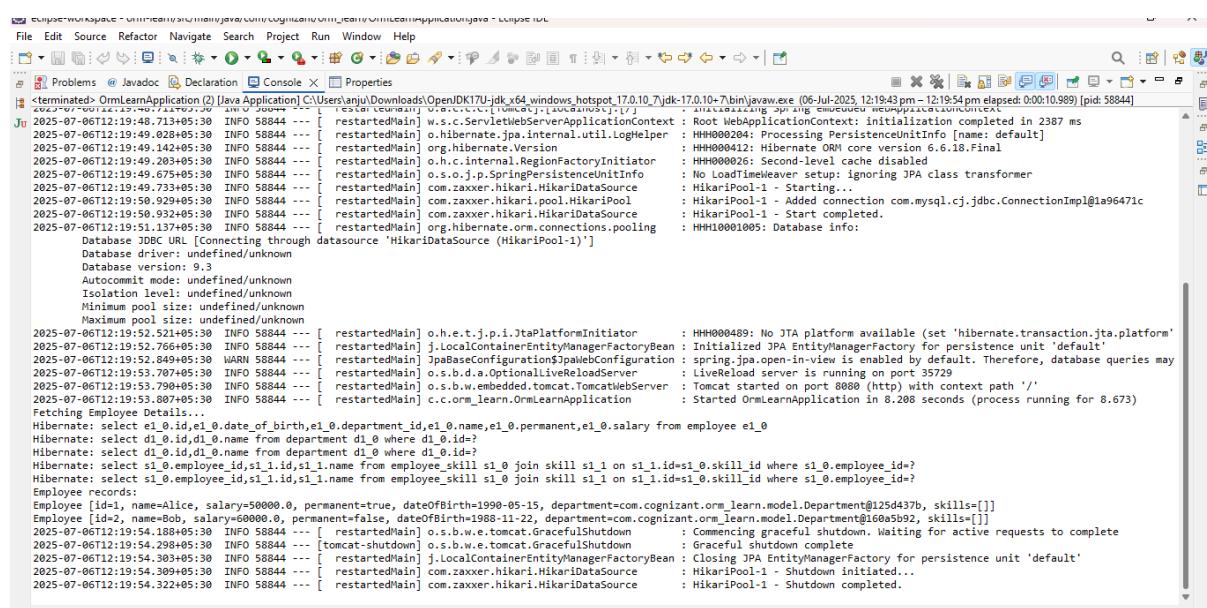
```
VALUES
```

```
('Alice Johnson', 75000, TRUE, '1990-03-25', 1),  
('Bob Smith', 60000, FALSE, '1992-07-15', 2),  
('Charlie Brown', 85000, TRUE, '1985-11-10', 3);
```

```
INSERT INTO employee_skill (employee_id, skill_id) VALUES
```

```
(1, 1),  
(1, 4),  
(2, 2),  
(2, 4),  
(3, 1),  
(3, 3);
```

Output:



```
File Edit Source Refactor Navigate Search Project Run Window Help  
Problems @ Javadoc Declaration Console Properties  
<terminated> OmLearnApplication [Java Application] C:\Users\anuj\Downloads\OpenDKT17u-jdk_x64_windows_hotspot_17.0.10_7\jdk-17.0.10\bin\javaw.exe (09-Jul-2023 12:19:43 pm) processId: 121954 pid: 58844 [id: 58844]  
2025-07-06T12:19:48.713+05:30 INFO 58844 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2387 ms  
2025-07-06T12:19:49.028+05:30 INFO 58844 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper :HHHH000204: Processing PersistenceUnitInfo [name: default]  
2025-07-06T12:19:49.142+05:30 INFO 58844 --- [ restartedMain] org.hibernate.Version : HHH0000412: Hibernate ORM core version 6.6.18.Final  
2025-07-06T12:19:49.203+05:30 INFO 58844 --- [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHHH000026: Second-level cache disabled  
2025-07-06T12:19:49.675+05:30 INFO 58844 --- [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInitializer : No LoadTimeWeaver setup: ignoring JPA class transformer  
2025-07-06T12:19:49.733+05:30 INFO 58844 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...  
2025-07-06T12:19:50.929+05:30 INFO 58844 --- [ restartedMain] com.zaxxer.hikari.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@1a96471c  
2025-07-06T12:19:50.932+05:30 INFO 58844 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.  
2025-07-06T12:19:51.137+05:30 INFO 58844 --- [ restartedMain] org.hibernate.orm.connections.pooling : HHHH0001005: Database info:  
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']  
Database driver: undefined/unknown  
Database version: 9.3  
Autocommit mode: undefined/unknown  
Isolation level: undefined/unknown  
Minimum pool size: undefined/unknown  
Max pool size: undefined/unknown  
Max total connections: undefined/unknown  
2025-07-06T12:19:52.521+05:30 INFO 58844 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform'  
2025-07-06T12:19:52.766+05:30 INFO 58844 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'  
2025-07-06T12:19:52.849+05:30 WARN 58844 --- [ restartedMain] jpaBaseConfiguration3pawebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may  
2025-07-06T12:19:53.707+05:30 INFO 58844 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729  
2025-07-06T12:19:53.790+05:30 INFO 58844 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8088 (http) with context path '/'  
2025-07-06T12:19:53.807+05:30 INFO 58844 --- [ restartedMain] c.c.orm_learnOrmLearnApplication : Started OmLearnApplication in 8.208 seconds (process running for 8.673)  
Fetching Employee Details...  
Hibernate: select e1_0.id,e1_0.date_of_birth,e1_0.department_id,e1_0.name,e1_0.permanent,e1_0.salary from employee e1_0  
Hibernate: select d1_0.id,d1_0.name from department d1_0 where d1_0.id=?  
Hibernate: select d1_0.id,d1_0.name from department d1_0 where d1_0.id=?  
Hibernate: select s1_0.employee_id,s1_1.id,s1_1.name from employee_skill s1_0 join skill s1_1 on s1_1.id=s1_0.skill_id where s1_0.employee_id=?  
Hibernate: select s1_0.employee_id,s1_1.id,s1_1.name from employee_skill s1_0 join skill s1_1 on s1_1.id=s1_0.skill_id where s1_0.employee_id=?  
Employee records:  
Employee [id=1, name=Alice, salary=50000.0, permanent=true, dateOfBirth=1998-05-15, department=com.cognizant.orm_learn.model.Department@125d437b, skills=[]]  
Employee [id=2, name=Bob, salary=60000.0, permanent=false, dateOfBirth=1988-11-22, department=com.cognizant.orm_learn.model.Department@160a5b92, skills=[]]  
2025-07-06T12:19:54.188+05:30 INFO 58844 --- [ restartedMain] o.s.b.w.e.tomcat.GracefulShutdown : Commencing graceful shutdown. Waiting for active requests to complete  
2025-07-06T12:19:54.209+05:30 INFO 58844 --- [tomcat-shutdown-thread-1] o.s.b.w.e.tomcat.GracefulShutdown : Graceful shutdown completed.  
2025-07-06T12:19:54.303+05:30 INFO 58844 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'  
2025-07-06T12:19:54.309+05:30 INFO 58844 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...  
2025-07-06T12:19:54.322+05:30 INFO 58844 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```

Implement many to one relationship between Employee and Department

```
package com.example.ormlearn;
```

```
import com.example.ormlearn.model.Department;
import com.example.ormlearn.model.Employee;
import com.example.ormlearn.service.DepartmentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.CommandLineRunner;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private DepartmentService departmentService;

    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        testGetDepartment();
    }

    private void testGetDepartment() {
        Department department = departmentService.get(1);
        if (department != null) {
            System.out.println("Department: " + department);
            System.out.println("Employees in Department:");
            for (Employee employee : department.getEmployeeList()) {
```

```
        System.out.println(employee);

    }

} else {

    System.out.println("Department not found!");

}

}

}
```

Ans

```
package com.example.ormlearn;

import com.example.ormlearn.model.Department;
import com.example.ormlearn.model.Employee;
import com.example.ormlearn.repository.DepartmentRepository;
import com.example.ormlearn.service.EmployeeService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

    private final EmployeeService employeeService;
```

```
private final DepartmentRepository departmentRepository;

public OrmLearnApplication(EmployeeService employeeService, DepartmentRepository departmentRepository) {

    this.employeeService = employeeService;
    this.departmentRepository = departmentRepository;
}

public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
}

@Override
public void run(String... args) {
    LOGGER.info("Start");

    Department department = new Department();
    department.setName("IT");
    departmentRepository.save(department);

    Employee employee = new Employee();
    employee.setName("John Doe");
    employee.setSalary(60000.0);
    employee.setPermanent(true);
    employee.setDepartment(department);
    employeeService.save(employee);

    Employee fetchedEmployee = employeeService.get(employee.getId());
    LOGGER.info("Fetched Employee: {}", fetchedEmployee);

    LOGGER.info("End");
}
}
```

```
package com.example.ormlearn.repository;

import com.example.ormlearn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Department;
import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}

package com.example.ormlearn;

import com.example.ormlearn.model.Department;
import com.example.ormlearn.model.Employee;
import com.example.ormlearn.repository.DepartmentRepository;
import com.example.ormlearn.service.EmployeeService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {
```

```
private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private final EmployeeService employeeService;
private final DepartmentRepository departmentRepository;

public OrmLearnApplication(EmployeeService employeeService, DepartmentRepository
departmentRepository) {
    this.employeeService = employeeService;
    this.departmentRepository = departmentRepository;
}

public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
}

@Override
public void run(String... args) {
    LOGGER.info("Start");
    Department department = new Department();
    department.setName("IT");
    departmentRepository.save(department);

    Employee employee = new Employee();
    employee.setName("John Doe");
    employee.setSalary(60000.0);
    employee.setPermanent(true);
    employee.setDepartment(department);
    employeeService.save(employee);

    Employee fetchedEmployee = employeeService.get(employee.getId());
    LOGGER.info("Fetched Employee: {}", fetchedEmployee);
}
```

```
    LOGGER.info("End");

}

}

package com.example.ormlearn.model;

import jakarta.persistence.*;

@Entity
@Table(name = "department")
public class Department {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
}
```

```
public void setName(String name) {  
    this.name = name;  
}  
  
@Override  
public String toString() {  
    return "Department [id=" + id + ", name=" + name + "]";  
}  
}  
  
package com.example.ormlearn.service;  
  
import com.example.ormlearn.model.Employee;  
import com.example.ormlearn.repository.EmployeeRepository;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
@Service  
public class EmployeeService {  
  
    private static final Logger LOGGER = LoggerFactory.getLogger(EmployeeService.class);  
  
    private final EmployeeRepository employeeRepository;  
  
    public EmployeeService(EmployeeRepository employeeRepository) {  
        this.employeeRepository = employeeRepository;  
    }  
  
    @Transactional
```

```
public Employee get(int id) {
    LOGGER.info("Fetching Employee with id: {}", id);
    return employeeRepository.findById(id).orElse(null);
}

@Transactional
public void save(Employee employee) {
    LOGGER.info("Saving Employee: {}", employee);
    employeeRepository.save(employee);
}

package com.example.ormlearn.service;

import com.example.ormlearn.model.Department;
import com.example.ormlearn.repository.DepartmentRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class DepartmentService {

    private final DepartmentRepository departmentRepository;

    public DepartmentService(DepartmentRepository departmentRepository) {
        this.departmentRepository = departmentRepository;
    }

    @Transactional
    public Department get(int id) {
        return departmentRepository.findById(id).orElse(null);
    }
}
```

```
@Transactional
```

```
public void save(Department department) {  
    departmentRepository.save(department);  
}
```

Output

```
Problems @ Javadoc Declaration Console X Properties  
OrmLearnApplication [1] [Java Application] C:\Users\anu\Downloads\OpenJDK17U-jdk-64-windows-hotspot_17.0.10_7\jdk-17.0.10_7\bin\javaw.exe (06-Jul-2025, 1:31:41 pm elapsed: 0:05:19) [pid: 111708]  
[main] o.apache.catalina.core.StandardService : Starting service [Tomcat]  
[main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]  
[main] o.a.c.c.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext  
[main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3818 ms  
[main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]  
[main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final  
[main] o.h.c.internal.RegionFactoryInitiator : HHH00026: Second-level cache disabled  
[main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer  
[main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting..  
[main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:test user=SA  
[main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.  
[main] org.hibernate.orm.connections.Pooling : HHH10001005: Database info:  
h datasource 'HikariDataSource (HikariPool-1)'  
  
-- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)  
er generated by default as identity, name varchar(255), primary key (id)  
generated by default as identity, name varchar(255), permanent boolean not null, salary float(53) not null, em_dp_id integer, primary key (id)  
d constraint FKpixkx4ywmn174m2c80tivxsks foreign key (em_dp_id) references department  
-- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'  
-- [main] jpaBaseConfigurationsJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicit:  
-- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:test'  
-- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''  
-- [main] c.example.ormlearnOrmLearnApplication : Started OrmLearnApplication in 9.554 seconds (process running for 10.663)  
-- [main] c.example.ormlearnOrmLearnApplication : Start  
values (?,default)  
-- [main] c.e.ormlearn.service.EmployeeService : Saving Employee: Employee [id=0, name=John Doe, salary=60000.0, permanent=true, department=Department [id=3, name=IT]]  
me,permanent,salary,id) values ( ?, ?, ?, ?)  
-- [main] c.e.ormlearn.service.EmployeeService : Fetching Employee with id: 2  
e1_0.name,e1_0.permanent,e1_0.salary from employee e1_0 left join department d1_0 on d1_0.id=e1_0.em_dp_id where e1_0.id=?  
-- [main] c.example.ormlearnOrmLearnApplication : Fetched Employee: Employee [id=2, name=John Doe, salary=60000.0, permanent=true, department=Department [id=3, name=IT]]  
-- [main] c.example.ormlearnOrmLearnApplication : End
```

Implement one to many relationship between Employee and Department

Answer:

```
package com.example.ormlearn.model;
```

```
import jakarta.persistence.*;
```

```
import java.util.List;
```

```
@Entity
```

```
@Table(name = "department")
```

```
public class Department {
```

```
    @Id
```

```
    private int id;
```

```
    private String name;
```

```
@OneToMany(mappedBy = "department", cascade = CascadeType.ALL, fetch = FetchType.EAGER)

private List<Employee> employeeList;

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public List<Employee> getEmployeeList() {
    return employeeList;
}

public void setEmployeeList(List<Employee> employeeList) {
    this.employeeList = employeeList;
}

@Override
public String toString() {
    return "Department [id=" + id + ", name=" + name + "]";
}
```

```
    }

}

package com.example.ormlearn;

import com.example.ormlearn.model.Department;
import com.example.ormlearn.model.Employee;
import com.example.ormlearn.service.DepartmentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.CommandLineRunner;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private DepartmentService departmentService;

    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        testGetDepartment();
    }

    private void testGetDepartment() {
        Department department = departmentService.get(1);
        if (department != null) {
```

```
        System.out.println("Department: " + department);
        System.out.println("Employees in Department:");
        for (Employee employee : department.getEmployeeList()) {
            System.out.println(employee);
        }
    } else {
        System.out.println("Department not found!");
    }
}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Department;
import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}

package com.example.ormlearn.service;

import com.example.ormlearn.model.Department;
import com.example.ormlearn.repository.DepartmentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class DepartmentService {

    @Autowired
    private DepartmentRepository departmentRepository;
```

```
public Department get(int id) {  
    Optional<Department> result = departmentRepository.findById(id);  
    return result.orElse(null);  
}  
}  
  
package com.example.ormlearn.model;  
  
  
import jakarta.persistence.*;  
  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
@Id  
@GeneratedValue(strategy = GenerationType.IDENTITY)  
private int id;  
  
  
private String name;  
private double salary;  
private boolean permanent;  
  
  
@ManyToOne  
@JoinColumn(name = "department_id")  
private Department department;  
  
  
public int getId() {  
    return id;  
}  
}
```

```
public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public boolean isPermanent() {
    return permanent;
}

public void setPermanent(boolean permanent) {
    this.permanent = permanent;
}

public Department getDepartment() {
    return department;
}
```

```

public void setDepartment(Department department) {
    this.department = department;
}

}

@Override
public String toString() {
    return "Employee [id=" + id + ", name=" + name + "]";
}

```

Output:

```

2025-07-06T14:11:47.018+05:30 INFO 122652 --- [main] c.example.ormlearn.OrmLearnApplication : Starting OrmLearnApplication using Java 17.0.10 with PID 122652 (C:\Users\anju\Downloads\OpenJDK17U-jdk_x64_windows_hotspot_17.0.10_7\jdk-17.0.10+7\bin\javaw.exe)
2025-07-06T14:11:47.037+05:30 INFO 122652 --- [main] c.example.ormlearn.OrmLearnApplication : Using profile: default, falling back to 1 default profile: "default"
2025-07-06T14:11:49.500+05:30 INFO 122652 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data repository in DEFAULT mode.
2025-07-06T14:11:49.791+05:30 INFO 122652 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 134 ms. Found 1 JPA repository int
2025-07-06T14:11:51.055+05:30 INFO 122652 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with repository scanning in 0 ms. Found 1 JPA repository int
2025-07-06T14:11:51.103+05:30 INFO 122652 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-06T14:11:51.183+05:30 INFO 122652 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-06T14:11:51.241+05:30 INFO 122652 --- [main] o.a.c.c.[Tomcat@localhost:[]] : Initializing Spring embedded WebApplicationContext
2025-07-06T14:11:51.244+05:30 INFO 122652 --- [main] w.s.c.WebApplicationContext : Root WebApplicationContext: initialization completed in 3921 ms
2025-07-06T14:11:51.840+05:30 INFO 122652 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-06T14:11:51.844+05:30 INFO 122652 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA
2025-07-06T14:11:51.844+05:30 INFO 122652 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-06T14:11:52.105+05:30 INFO 122652 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-06T14:11:52.236+05:30 INFO 122652 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-06T14:11:52.315+05:30 INFO 122652 --- [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2025-07-06T14:11:52.889+05:30 INFO 122652 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-06T14:11:53.065+05:30 INFO 122652 --- [main] org.hibernate.orm.connections.pooling : HHH1001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 2.3.3
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
Hibernate: alter table if exists department alter column name set data type varchar(255)
Hibernate: alter table if exists employee alter column name set data type varchar(255)
2025-07-06T14:11:54.823+05:30 INFO 122652 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-06T14:11:55.132+05:30 WARN 122652 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may
2025-07-06T14:11:55.697+05:30 INFO 122652 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-06T14:11:55.713+05:30 INFO 122652 --- [main] c.example.ormlearn.OrmLearnApplication : Started OrmLearnApplication in 10.242 seconds (process running for 10.989)
Hibernate: select d1_0.id,d1_0.name,e11_0.department_id,e11_0.id,e11_0.name,e11_0.persistent,e11_0.salary from department d1_0 left join employee e11_0 on d1_0.id=e11_0.department_id where
Department: Department [id=1, name=Development]
Employees in Department:
Employee [id=1, name=John Doe]
Employee [id=2, name=Jane Smith]

```

Implement many to many relationship between Employee and Skill

```

package com.example.ormlearn;

import com.example.ormlearn.model.Employee;
import com.example.ormlearn.model.Skill;
import com.example.ormlearn.service.EmployeeService;
import com.example.ormlearn.service.SkillService;
import org.springframework.beans.factory.annotation.Autowired;

```

```
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private EmployeeService employeeService;

    @Autowired
    private SkillService skillService;

    private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        testAddSkillToEmployee();
    }

    public void testAddSkillToEmployee() {
        Employee employee = employeeService.get(1);
        Skill skill = skillService.get(2);

        if (employee != null && skill != null) {
```

```
employee.getSkillList().add(skill);
employeeService.save(employee);

LOGGER.debug("Skill added successfully to employee");
System.out.println("Skill added successfully to employee");
System.out.println("Updated Employee: " + employee);
System.out.println("Skills: " + employee.getSkillList());

} else {
LOGGER.debug("Employee or Skill not found");
System.out.println("Employee or Skill not found");
}

}

}

package com.example.ormlearn.model;

import jakarta.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    private int id;
    private String name;
    private double salary;
    private boolean permanent;

    @ManyToOne
    @JoinColumn(name = "department_id")
}
```

```

private Department department;

@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(name = "employee_skill",
    joinColumns = @JoinColumn(name = "es_em_id"),
    inverseJoinColumns = @JoinColumn(name = "es_sk_id"))

private Set<Skill> skillList = new HashSet<>();

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department = department; }

public Set<Skill> getSkillList() { return skillList; }

public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

@Override

public String toString() {

    return "Employee{" +
        "id=" + id +
        ", name='" + name + '\'' +
        ", salary=" + salary +
        ", permanent=" + permanent +
        ", department=" + (department != null ? department.getName() : "null") +
        '}';
}

```

```
}
```

```
package com.example.ormlearn.model;
```

```
import jakarta.persistence.*;
```

```
import java.util.HashSet;
```

```
import java.util.Set;
```

```
@Entity
```

```
@Table(name = "skill")
```

```
public class Skill {
```

```
    @Id
```

```
    private int id;
```

```
    private String name;
```

```
    @ManyToMany(mappedBy = "skillList")
```

```
    private Set<Employee> employeeList = new HashSet<>();
```

```
    public int getId() { return id; }
```

```
    public void setId(int id) { this.id = id; }
```

```
    public String getName() { return name; }
```

```
    public void setName(String name) { this.name = name; }
```

```
    public Set<Employee> getEmployeeList() { return employeeList; }
```

```
    public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Skill{" +
```

```
            "id=" + id +
```

```

        ", name='" + name + "\\"
    '}';
}

}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Skill;
import org.springframework.data.jpa.repository.JpaRepository;

public interface SkillRepository extends JpaRepository<Skill, Integer> {
}

```

Output:

```

Problems Javadoc Declaration Console Properties
OrmLearnApplication (1) [Java Application] C:\Users\Anuj\Downloads\OpenJDK17U-jdk-x64_windows_hotspot_17.0.10_7\jdk-17.0.10+7\bin\javaw.exe (06-Jul-2025, 2:37:18 pm elapsed: 0:02:15) [pid: 130984]
2025-07-06T14:37:23.176+05:30 INFO 130984 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet Engine: [Apache Tomcat/10.1.42]
2025-07-06T14:37:23.295+05:30 INFO 130984 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-07-06T14:37:23.297+05:30 INFO 130984 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2953 ms
2025-07-06T14:37:23.477+05:30 INFO 130984 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-06T14:37:23.851+05:30 INFO 130984 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA
2025-07-06T14:37:23.855+05:30 INFO 130984 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-06T14:37:24.094+05:30 INFO 130984 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-06T14:37:24.210+05:30 INFO 130984 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-06T14:37:24.279+05:30 INFO 130984 --- [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2025-07-06T14:37:24.838+05:30 INFO 130984 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-06T14:37:25.009+05:30 INFO 130984 --- [main] org.hibernate.orm.connections.Pool : HHHM10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 2.3.232
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-06T14:37:26.493+05:30 INFO 130984 --- [main] o.h.e.t.j.p.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform')
Hibernate: alter table if exists department alter column name set data type varchar(255)
Hibernate: alter table if exists employee alter column name set data type varchar(255)
Hibernate: alter table if exists skill alter column name set data type varchar(255)
2025-07-06T14:37:26.579+05:30 INFO 130984 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-06T14:37:26.965+05:30 WARN 130984 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be executed via JPA queries (e.g. EntityManager.find(...)). This leads to second-level cache being disabled. To prevent this, configure spring.jpa.open-in-view=false or use a different configuration mechanism.
2025-07-06T14:37:27.708+05:30 INFO 130984 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-06T14:37:27.725+05:30 INFO 130984 --- [main] c.example.ormlearnOrmLearnApplication : Started OrmLearnApplication in 7.982 seconds (process running for 8.52)
Hibernate: select e1_0.id,d1_0.id,d1_0.name,e1_0.name,e1_0.persistent,e1_0.salary,s1_0.es_em_id,s1_1.id,s1_1.name from employee e1_0 left join department d1_0 on d1_0.id=e1_0.department
Hibernate: select e1_0.id,d1_0.id,e1_0.name,e1_0.persistent,e1_0.salary,s1_0.es_em_id,s1_1.id,s1_1.name from employee e1_0 left join department d1_0 on d1_0.id=e1_0.department
Hibernate: select s1_0.id,s1_0.name from skill s1_0 where e1_0.id=?
Hibernate: select e1_0.id,d1_0.id,d1_0.name,e1_0.name,e1_0.persistent,e1_0.salary,s1_0.es_em_id,s1_1.id,s1_1.name from employee e1_0 left join department d1_0 on d1_0.id=e1_0.department
Hibernate: select e1_0.department_id,e1_0.id,e1_0.name,e1_0.persistent,e1_0.salary from employee e1_0 where e1_0.department_id?
Hibernate: select s1_0.id,s1_0.name from skill s1_0 where e1_0.id?
Hibernate: insert into employee_skill (es_em_id,es_sk_id) values (?,?)
Skill added successfully to employee
Updated Employee: Employee{id=1, name='John Doe', salary=50000.0, permanent=true, department=IT}
Skills: [Skill{id=2, name='Spring Boot'}]

```

Get all permanent employees using HQL

Answer:

```
package com.example.ormlearn.service;

import com.example.ormlearn.model.Employee;
import com.example.ormlearn.model.Skill;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;

@Service
public class TestEmployeeService {

    @Autowired
    private EmployeeService employeeService;

    @Autowired
    private SkillService skillService;

    private static final Logger LOGGER = LoggerFactory.getLogger(TestEmployeeService.class);

    @Transactional
    public void addSkillToEmployee() {
        Employee employee = employeeService.get(1);
        Skill skill = skillService.get(2);

        if (employee != null && skill != null) {
            if (!employee.getSkillList().contains(skill)) {
                employee.getSkillList().add(skill);
            }
        }
    }
}
```

```
employeeService.save(employee);

    LOGGER.info("Skill '{}' added successfully to employee '{}'", skill.getName(),
employee.getName());

} else {

    LOGGER.info("Skill '{}' already exists for employee '{}'", skill.getName(),
employee.getName());

}

} else {

    LOGGER.warn("Employee or Skill not found for adding skill.");
}

}

}

@Transactional
public void fetchAllPermanentEmployees() {

    LOGGER.info("Fetching all permanent employees... ");

    List<Employee> employees = employeeService.getAllPermanentEmployees();

    for (Employee employee : employees) {

        String departmentName = (employee.getDepartment() != null) ?
employee.getDepartment().getName() : "No Department";

        LOGGER.info("Employee: {}, Department: {}, Skills: {}",
employee.getName(),
departmentName,
employee.getSkillList());
    }

    LOGGER.info("Completed fetching permanent employees.");
}

}

package com.example.ormlearn.model;

import jakarta.persistence.*;
```

```
@Entity
@Table(name = "department")
public class Department {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

package com.example.ormlearn.model;

import jakarta.persistence.*;
import java.util.HashSet;
```

```
import java.util.Set;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    private boolean permanent;

    private double salary;

    @ManyToOne
    @JoinColumn(name = "em_dp_id")
    private Department department;

    @ManyToMany
    @JoinTable(
        name = "employee_skill",
        joinColumns = @JoinColumn(name = "es_em_id"),
        inverseJoinColumns = @JoinColumn(name = "es_sk_id")
    )
    private Set<Skill> skillList = new HashSet<>();

    public int getId() {
        return id;
    }
}
```

```
public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public boolean isPermanent() {
    return permanent;
}

public void setPermanent(boolean permanent) {
    this.permanent = permanent;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public Department getDepartment() {
    return department;
}
```

```
}

public void setDepartment(Department department) {
    this.department = department;
}

public Set<Skill> getSkillList() {
    return skillList;
}

public void setSkillList(Set<Skill> skillList) {
    this.skillList = skillList;
}

package com.example.ormlearn.model;

import jakarta.persistence.*;
import java.util.Objects;

@Entity
@Table(name = "skill")
public class Skill {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    public int getId() {
        return id;
    }
}
```

```
}
```

```
public void setId(int id) {
```

```
    this.id = id;
```

```
}
```

```
public String getName() {
```

```
    return name;
```

```
}
```

```
public void setName(String name) {
```

```
    this.name = name;
```

```
}
```

```
@Override
```

```
public String toString() {
```

```
    return name;
```

```
}
```

```
@Override
```

```
public boolean equals(Object o) {
```

```
    if (this == o) return true;
```

```
    if (!(o instanceof Skill)) return false;
```

```
    Skill skill = (Skill) o;
```

```
    return id == skill.id;
```

```
}
```

```
@Override
```

```
public int hashCode() {
```

```
    return Objects.hash(id);
```

```
}
```

```

}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

    @Query("SELECT e FROM Employee e " +
            "LEFT JOIN FETCH e.department d " +
            "LEFT JOIN FETCH e.skillList " +
            "WHERE e.permanent = true")
    List<Employee> getAllPermanentEmployees();

}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Skill;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface SkillRepository extends JpaRepository<Skill, Integer> {
}

Output:

```

```

:---[main] c.e.o.service.TestEmployeeService : Skill 'Python' already exists for employee 'John Doe'
:---[main] c.e.o.service.TestEmployeeService : Fetching all permanent employees...
:me,e1_0.name,e1_0.permanent,e1_0.salary,s11_0.es_em_id,s11_0.id,s11_0.name : Employee e1_0 left join department d1_0 on d1_0.id=e1_0.em_dp_id left join employee_skill s11_0 on e1_0.i
:---[main] c.e.o.service.TestEmployeeService : Employee: John Doe, Department: IT, Skills: [Java, Python, SQL]
:---[main] c.e.o.service.TestEmployeeService : Employee: Mark Johnson, Department: HR, Skills: [Java, Python]
:---[main] c.e.o.service.TestEmployeeService : Completed fetching permanent employees.

```

Fetch Quiz Attemp Details

Answer:

```
package com.example.ormlearn;

import com.example.ormlearn.model.Attempt;
import com.example.ormlearn.model.AttemptQuestion;
import com.example.ormlearn.repository.AttemptRepository;
import com.example.ormlearn.repository.AttemptQuestionRepository;
import com.example.ormlearn.service.AttemptService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.sql.Timestamp;
import java.time.LocalDateTime;
import java.util.Arrays;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private AttemptService attemptService;

    @Autowired
    private AttemptRepository attemptRepository;

    @Autowired
    private AttemptQuestionRepository attemptQuestionRepository;
```

```
public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
}

@Override
public void run(String... args) throws Exception {
    Attempt attempt = attemptRepository.findByUserAndId("john_doe", 1);
    if (attempt == null) {
        attempt = new Attempt();
        attempt.setUser("john_doe");
        attempt.setDate(Timestamp.valueOf(LocalDateTime.now()));
        attempt = attemptRepository.save(attempt);

        AttemptQuestion q1 = new AttemptQuestion();
        q1.setAttempt(attempt);
        q1.setQuestionText("What is Java?");

        AttemptQuestion q2 = new AttemptQuestion();
        q2.setAttempt(attempt);
        q2.setQuestionText("Explain Hibernate.");

        attemptQuestionRepository.saveAll(Arrays.asList(q1, q2));

        System.out.println(" Test data inserted with Attempt ID: " + attempt.getId());
    }

    System.out.println("\n All Attempts in DB:");
    attemptRepository.findAll().forEach(a -> {
        System.out.println("Attempt ID: " + a.getId() + ", User: " + a.getUser());
    });
}
```

```
Attempt fetchedAttempt = attemptService.getAttempt("john_doe", attempt.getId());
if (fetchedAttempt != null) {
    System.out.println("\n Attempt found for user: " + fetchedAttempt.getUser());
    System.out.println("Attempt Date: " + fetchedAttempt.getDate());
    fetchedAttempt.getQuestions().forEach(q ->
        System.out.println("Question: " + q.getQuestionText())
    );
} else {
    System.out.println("Attempt not found");
}
}

}

package com.example.ormlearn.model;

import jakarta.persistence.*;
import java.sql.Timestamp;
import java.util.List;

@Entity
@Table(name = "attempt")
public class Attempt {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String user;

    private Timestamp date;
```

```
@OneToOne(mappedBy = "attempt", cascade = CascadeType.ALL, fetch = FetchType.LAZY)

private List<AttemptQuestion> questions;

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getUser() {
    return user;
}

public void setUser(String user) {
    this.user = user;
}

public Timestamp getDate() {
    return date;
}

public void setDate(Timestamp date) {
    this.date = date;
}

public List<AttemptQuestion> getQuestions() {
    return questions;
}
```

```
public void setQuestions(List<AttemptQuestion> questions) {  
    this.questions = questions;  
}  
}
```

```
package com.example.ormlearn.model;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
public class AttemptOption {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "attempt_question_id")
```

```
    private AttemptQuestion attemptQuestion;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "option_id")
```

```
    private Option option;
```

```
    public Long getId() {
```

```
        return id;
```

```
    }
```

```
    public void setId(Long id) {
```

```
        this.id = id;
```

```
    }
```

```
public AttemptQuestion getAttemptQuestion() {
    return attemptQuestion;
}

public void setAttemptQuestion(AttemptQuestion attemptQuestion) {
    this.attemptQuestion = attemptQuestion;
}

public Option getOption() {
    return option;
}

public void setOption(Option option) {
    this.option = option;
}

package com.example.ormlearn.model;

import jakarta.persistence.*;

@Entity
@Table(name = "attempt_question")
public class AttemptQuestion {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @ManyToOne
    @JoinColumn(name = "attempt_id")
```

```
private Attempt attempt;

private String questionText;

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public Attempt getAttempt() {
    return attempt;
}

public void setAttempt(Attempt attempt) {
    this.attempt = attempt;
}

public String getQuestionText() {
    return questionText;
}

public void setQuestionText(String questionText) {
    this.questionText = questionText;
}

package com.example.ormlearn.model;

import jakarta.persistence.*;
```

```
@Entity  
public class Option {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String text;  
  
    private int score;  
  
    @ManyToOne  
    @JoinColumn(name = "question_id")  
    private Question question;  
  
    public Long getId() {  
        return id;  
    }  
  
    public void setId(Long id) {  
        this.id = id;  
    }  
  
    public String getText() {  
        return text;  
    }  
  
    public void setText(String text) {  
        this.text = text;  
    }  
}
```

```
public int getScore() {  
    return score;  
}  
  
public void setScore(int score) {  
    this.score = score;  
}  
  
public Question getQuestion() {  
    return question;  
}  
  
public void setQuestion(Question question) {  
    this.question = question;  
}  
}  
  
package com.example.ormlearn.model;  
  
import jakarta.persistence.*;  
import java.util.Set;  
  
@Entity  
public class Question {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String text;
```

```
@OneToOne(mappedBy = "question", cascade = CascadeType.ALL)
private Set<Option> options;

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

public Set<Option> getOptions() {
    return options;
}

public void setOptions(Set<Option> options) {
    this.options = options;
}

package com.example.ormlearn.model;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
@Table(name = "user")
```

```
public class User {
```

```
    @Id
```

```
    private int id;
```

```
    private String name;
```

```
    public int getId() {
```

```
        return id;
```

```
}
```

```
    public String getName() {
```

```
        return name;
```

```
}
```

Output:

```
maximum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-06T18:43:54.398+05:30 INFO 1300880 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator      : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to 'true')
2025-07-06T18:43:54.405+05:30 INFO 1300880 --- [main] j.LocalContainerEntityManagerFactoryBean   : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-06T18:43:55.621+05:30 INFO 1300880 --- [main] o.s.d.j.r.QueryEnhancerFactory          : Hibernate is in classpath; If applicable, HQL parser will be used.
2025-07-06T18:43:57.734+05:30 WARN 1300880 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during a view's lifecycle. See https://vladmihalcea.com/about-spring-data-jpa-open-in-view/
2025-07-06T18:43:58.478+05:30 INFO 1300880 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-06T18:43:58.499+05:30 INFO 1300880 --- [main] c.example.ormlearnOrmLearnApplication : Started OrmLearnApplication in 14.288 seconds (process running for 16.884)
Hibernate: select a1_0.id,a1_0.date,a1_0.user from attempt a1_0 where a1_0.user=? and a1_0.id=?
Hibernate: insert into attempt (date,user) values (?,?)
Hibernate: insert into attempt_question (attempt_id,question_text) values (?,?)
Hibernate: insert into attempt_question (attempt_id,question_text) values (?,?)
 Test data inserted with Attempt ID: 4
 All Attempts in DB:
Hibernate: select a1_0.id,a1_0.date,a1_0.user from attempt a1_0
Attempt ID: 1, User: testuser
Attempt ID: 2, User: john_doe
Attempt ID: 3, User: john_doe
Attempt ID: 4, User: john_doe
Hibernate: select a1_0.id,a1_0.date,q1_0.attempt_id,q1_0.id,q1_0.question_text,a1_0.user from attempt a1_0 join attempt_question q1_0 on a1_0.id=q1_0.attempt_id where a1_0.user=? and a1_
 Attempt found for user: john_doe
Attempt Date: 2025-07-06 10:43:59.0
Question: What is Java?
Question: Explain Hibernate.
```

Get average salary using HQL

Answer

```
package com.example.ormlearn;
```

```
import com.example.ormlearn.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private EmployeeService employeeService;

    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        int departmentId = 1;
        double avgSalary = employeeService.getAverageSalary(departmentId);
        System.out.println("Average salary for Department ID " + departmentId + " is: " + avgSalary);
    }
}

package com.example.ormlearn.model;

import jakarta.persistence.*;
```

```
@Id  
@GeneratedValue(strategy = GenerationType.IDENTITY)  
private int id;  
  
private String name;  
public int getId() {  
    return id;  
}  
  
public void setId(int id) {  
    this.id = id;  
}  
  
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
}  
  
package com.example.ormlearn.model;  
  
import jakarta.persistence.*;  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private double salary;

@ManyToOne
@JoinColumn(name = "department_id")
private Department department;

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
```

```
this.salary = salary;
}

public Department getDepartment() {
    return department;
}

public void setDepartment(Department department) {
    this.department = department;
}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

    @Query("SELECT AVG(e.salary) FROM Employee e WHERE e.department.id = :id")
    double getAverageSalary(@Param("id") int id);
}

Output:
```

```

:: Spring Boot ::      (v3.5.3)

2025-07-06T19:16:19.047+05:30 INFO 137064 --- [main] c.example.ormlearnOrmLearnApplication : Starting OrmLearnApplication using Java 17.0.10 with PID 137064 (C:\Users\anju\Downloads\OpenJDK17u-jdk_x64_windows_hotspot_17.0.10_7\bin\javaw.exe)
2025-07-06T19:16:19.055+05:30 INFO 137064 --- [main] c.example.ormlearnOrmLearnApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-06T19:16:20.474+05:30 INFO 137064 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-06T19:16:21.694+05:30 INFO 137064 --- [main] s.o.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-07-06T19:16:21.736+05:30 INFO 137064 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-06T19:16:21.737+05:30 INFO 137064 --- [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2025-07-06T19:16:21.737+05:30 INFO 137064 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2690 ms
2025-07-06T19:16:22.155+05:30 INFO 137064 --- [main] o.hibernate.jn.internal.util.LogHelper : HHH000042: Hibernate core version: 6.6.18.Final
2025-07-06T19:16:22.253+05:30 INFO 137064 --- [main] org.hibernate.Version : HHH000041: Hibernate Validator 0.14.0
2025-07-06T19:16:22.339+05:30 INFO 137064 --- [main] org.hibernate.orm.connections.Pool : HHH000026: Second level cache disabled
2025-07-06T19:16:22.808+05:30 INFO 137064 --- [main] o.s.o.j.p.SpringPersistenceInitInfo : No LoadTimeWeaver setup; ignoring JPA class transformer
2025-07-06T19:16:22.856+05:30 INFO 137064 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-06T19:16:23.465+05:30 INFO 137064 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@21b744ec
2025-07-06T19:16:23.469+05:30 INFO 137064 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-06T19:16:23.660+05:30 INFO 137064 --- [main] org.hibernate.orm.connections.Pool : HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 9.3
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-06T19:16:25.476+05:30 INFO 137064 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform')
2025-07-06T19:16:25.476+05:30 INFO 137064 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-06T19:16:25.978+05:30 INFO 137064 --- [main] o.s.d.j.r.QueryEnhancerFactory : Hibernate is in classpath; if applicable, JSQL parser will be used.
2025-07-06T19:16:27.142+05:30 WARN 137064 --- [main] jpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed via JSF or equivalent
2025-07-06T19:16:27.866+05:30 INFO 137064 --- [main] o.s.o.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-06T19:16:27.892+05:30 INFO 137064 --- [main] c.example.ormlearnOrmLearnApplication : Started OrmLearnApplication in 9.848 seconds (process running for 10.432)
Hibernate: select avg(e1_0.salary) from employee e1_0 where e1_0.department_id=?
Average salary for Department ID 1 is: 52500.0

```

Get all employees using Native Query

Answer:

```
package com.example.ormlearn;
```

```
import com.example.ormlearn.model.Employee;
import com.example.ormlearn.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import java.util.List;
```

```
@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {
```

```
@Autowired
```

```
private EmployeeService employeeService;
```

```
public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
}

@Override
public void run(String... args) throws Exception {
    System.out.println("All Employees (Using Native Query):");
    List<Employee> employees = employeeService.getAllEmployeesNative();
    employees.forEach(System.out::println);
}
}

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

    @Query(value = "SELECT * FROM employee", nativeQuery = true)
    List<Employee> getAllEmployeesNative();
}

package com.example.ormlearn.service;

import com.example.ormlearn.model.Employee;
import com.example.ormlearn.repository.EmployeeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
```

```
@Service
```

```
public class EmployeeService {
```

```
@Autowired
```

```
private EmployeeRepository employeeRepository;
```

```
public List<Employee> getAllEmployeesNative() {
```

```
    return employeeRepository.getAllEmployeesNative();
```

```
}
```

Output:

```
=====
:: Spring Boot ::          (v3.5.3)

2025-07-06T19:16:19.047+05:30 INFO 137064 --- [main] c.example.ormlearn.OrmLearnApplication : Starting OrmLearnApplication using Java 17.0.10 with PID 137064 (C:\Users\anji
2025-07-06T19:16:19.055+05:30 INFO 137064 --- [main] c.example.ormlearn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-06T19:16:20.474+05:30 INFO 137064 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-06T19:16:20.584+05:30 INFO 137064 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 88 ms. Found 1 JPA repository int
2025-07-06T19:16:21.694+05:30 INFO 137064 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-07-06T19:16:21.736+05:30 INFO 137064 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-06T19:16:21.737+05:30 INFO 137064 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-06T19:16:21.847+05:30 INFO 137064 --- [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2025-07-06T19:16:21.849+05:30 INFO 137064 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2690 ms
2025-07-06T19:16:22.155+05:30 INFO 137064 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-06T19:16:22.253+05:30 INFO 137064 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-06T19:16:22.328+05:30 INFO 137064 --- [main] o.h.c.internal.RegionFactoryInitiator : HHH000413: Second-level cache disabled
2025-07-06T19:16:22.356+05:30 INFO 137064 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : HHH000414: New configuration ignoring JPA class transformer
2025-07-06T19:16:22.456+05:30 INFO 137064 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-06T19:16:23.465+05:30 INFO 137064 --- [main] com.zaxxer.hikari.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@21b744ec
2025-07-06T19:16:23.469+05:30 INFO 137064 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-06T19:16:23.660+05:30 INFO 137064 --- [main] org.hibernate.orm.connections.Pooling : HHH10001095: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 9.3
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-06T19:16:25.476+05:30 INFO 137064 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform'
2025-07-06T19:16:25.481+05:30 INFO 137064 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-06T19:16:25.974+05:30 INFO 137064 --- [main] o.s.d.j.r.query.QueryEnhancerFactory : Hibernate is in classpath; If applicable, HQL parser will be used.
2025-07-06T19:16:27.142+05:30 WARN 137064 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may
2025-07-06T19:16:27.866+05:30 INFO 137064 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-07-06T19:16:27.892+05:30 INFO 137064 --- [main] c.example.ormlearn.OrmLearnApplication : Started OrmLearnApplication in 9.848 seconds (process running for 10.432)
Hibernate: select avg(e1_0.salary) from employee e1_0 where e1_0.department_id=?
Average salary for Department ID 1 is: 52500.0
```

