**Hands on 1**

**Spring Data JPA - Quick Example**

**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;

@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);

LOGGER.info("Inside main");

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;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "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;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

CountryService.java

package com.cognizant.ormlearn;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**application.properties**

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

# Console log pattern

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# MySQL DB configuration

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.18</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.0.33</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**SQL for MySQL Table**

CREATE DATABASE IF NOT EXISTS ormlearn;

USE ormlearn;

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

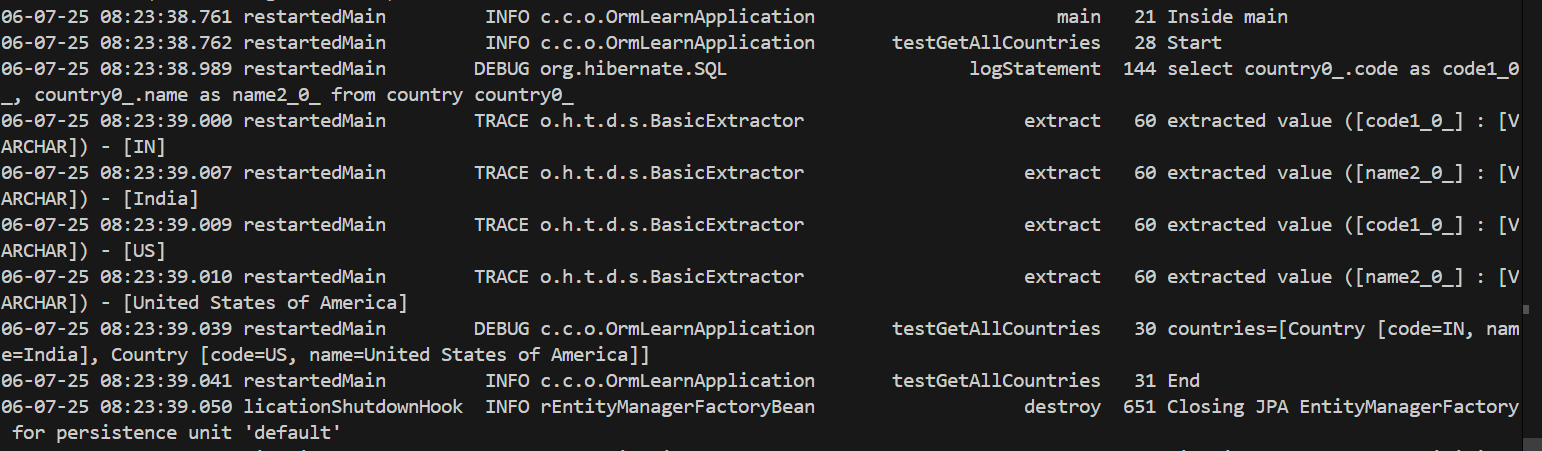
name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

**Output:**



**Hands on 2**

**Hibernate XML Config implementation walk through** 

Object-Relational Mapping (ORM) via XML

In Hibernate, the **object (Java class)** is mapped to a **relational database table** using a hbm.xml mapping file.

**Example** – Employee.hbm.xml:

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.tutorialspoint.Employee" table="EMPLOYEE">

<id name="id" type="int" column="id">

<generator class="native"/>

</id>

<property name="firstName" column="first\_name" type="string"/>

<property name="lastName" column="last\_name" type="string"/>

<property name="salary" column="salary" type="float"/>

</class>

</hibernate-mapping>

It maps:

* Java class Employee to table EMPLOYEE
* firstName, lastName, salary fields to their respective columns

Hibernate Configuration

hibernate.cfg.xml

This file configures DB connection settings and includes mapping files.

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernatedb</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="show\_sql">true</property>

<property name="hbm2ddl.auto">update</property>

<mapping resource="Employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**Key Hibernate Concepts with Examples**

**🔸 SessionFactory**

* It is a heavyweight object used to create Session objects.
* It is created once during app startup.

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.buildSessionFactory();

**🔸 Session**

* It represents a connection between the Java app and the database.
* Used to perform CRUD operations.

Session session = factory.openSession();

**🔸 Transaction**

* Represents a unit of work.
* Ensures **atomicity** of DB operations.

Transaction tx = session.beginTransaction();

**🔸 beginTransaction()**

* Begins the transaction.

Transaction tx = session.beginTransaction();

**🔸 commit()**

* Commits the transaction — changes are saved in DB.

tx.commit();

**🔸 rollback()**

* Undoes the transaction if an error occurs.

tx.rollback();

**🔸 session.save()**

* Inserts a new record in the DB

Employee emp = new Employee();

emp.setFirstName("John");

emp.setLastName("Doe");

emp.setSalary(50000);

session.save(emp);

**🔸 session.createQuery().list()**

* Executes HQL (Hibernate Query Language) and returns results.

List<Employee> employees = session.createQuery("FROM Employee").list();

**🔸 session.get()**

* Retrieves an object by primary key.

Employee emp = session.get(Employee.class, 1);

**🔸 session.delete()**

* Deletes a row in the database.

Employee emp = session.get(Employee.class, 1);

session.delete(emp);

Example: Full Flow

public class MainApp {

public static void main(String[] args) {

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee emp = new Employee();

emp.setFirstName("Yashaswini");

emp.setLastName("K");

emp.setSalary(60000);

session.save(emp);

List<Employee> list = session.createQuery("FROM Employee").list();

for (Employee e : list) {

System.out.println(e);

}

tx.commit();

} catch (Exception e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}

**Hands on 3**

**Hibernate Annotation Config implementation walk through** 

**Object-Relational Mapping using Annotations in Hibernate**

Instead of using .hbm.xml files, Hibernate allows mapping Java classes to database tables **directly using annotations** in the entity class.

**Persistence Class: Employee.java**

This class maps to the EMPLOYEE table.

import javax.persistence.\*;

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "salary")

private int salary;

}

Explanation of Annotations

| **Annotation** | **Purpose** |
| --- | --- |
| @Entity | Declares the class as a Hibernate entity (persistent class) |
| @Table(name=...) | Maps the class to a specific DB table |
| @Id | Declares the primary key of the entity |
| @GeneratedValue | Auto-generates primary key values (can use strategy like AUTO, IDENTITY) |
| @Column | Maps the field to a specific column in the table |

**Hibernate Configuration (hibernate.cfg.xml)**

This XML file configures Hibernate’s connection to the database and specifies annotated classes.

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernatedb</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="show\_sql">true</property>

<property name="hbm2ddl.auto">update</property>

<mapping class="com.tutorialspoint.Employee"/>

</session-factory>

</hibernate-configuration>

**Configuration Details**

| **Property** | **Description** |
| --- | --- |
| hibernate.dialect | Tells Hibernate how to convert HQL to SQL for a specific DB (e.g., MySQL) |
| hibernate.connection.driver\_class | JDBC driver for the DB (e.g., MySQL Connector/J) |
| hibernate.connection.url | JDBC URL to connect to the DB |
| hibernate.connection.username/password | DB credentials |

**Example: Insert Employee**

public class MainApp {

public static void main(String[] args) {

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

Session session = factory.openSession();

try {

Employee emp = new Employee();

emp.setFirstName("Yashaswini");

emp.setLastName("Kurapati");

emp.setSalary(75000);

session.beginTransaction();

session.save(emp);

session.getTransaction().commit();

System.out.println("Inserted: " + emp.getId());

} finally {

session.close();

factory.close();

}

}

}

**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA**   
  
**1. Java Persistence API (JPA)**

| **Feature** | **Description** |
| --- | --- |
| Definition | JPA is a **specification** (JSR 338) for ORM (Object Relational Mapping) in Java. |
| Purpose | Provides a standard to map Java objects to relational databases. |
| Key Point | **JPA is only a set of interfaces**—it does not implement anything itself. |
| Popular Implementations | Hibernate, EclipseLink, OpenJPA |

**2. Hibernate**

| **Feature** | **Description** |
| --- | --- |
| Definition | Hibernate is a **concrete implementation** of JPA. |
| Type | A full-featured ORM framework. |
| Key Features | Lazy loading, caching, transaction management, HQL |
| Usage | You use SessionFactory, Session, and Transaction manually. |

**Example: Hibernate Insert 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;

}

**3. Spring Data JPA**

| **Feature** | **Description** |
| --- | --- |
| Definition | Spring Data JPA is a **wrapper around JPA/Hibernate** |
| Type | Provides a higher-level abstraction over JPA/Hibernate |
| Purpose | Reduces boilerplate code (no need to manage Session or write queries manually) |
| Key Features | Built-in CRUD methods, Custom queries, Paging & Sorting, Transaction Management |

**Example: Spring Data JPA Insert Code**

**EmployeeRepository.java**

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

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**Comparison**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (Interface) | Implementation | Abstraction Layer |
| Provides ORM? | No | Yes | Uses Hibernate under the hood |
| Boilerplate Code | Requires | More | Minimal |
| Transaction Management | Manual (JPA APIs) | Manual | Handled by Spring @Transactional |
| CRUD Operations | Must implement | Must implement | Provided by JpaRepository |
| Popular Usage | Used via implementation | Used directly or via JPA | Widely used with Spring Boot |

**Hands on 5**

**Implement services for managing Country**   
  
**Country.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

private String name;

public Country() {

}

public Country(String code, String name) {

this.code = code;

this.name = 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 java.util.List;

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

import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String keyword);

}

**CountryService.java**

package com.cognizant.ormlearn.service;

import java.util.List;

import com.cognizant.ormlearn.model.Country;

public interface CountryService {

List<Country> getAllCountries();

Country findCountryByCode(String code);

void addCountry(Country country);

void updateCountry(Country country);

void deleteCountry(String code);

List<Country> searchCountries(String keyword);

}

**CountryServiceImpl.java**

package com.cognizant.ormlearn.service;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

import jakarta.transaction.Transactional;

@Service

public class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository countryRepository;

@Override

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

@Override

public Country findCountryByCode(String code) {

Optional<Country> country = countryRepository.findById(code);

return country.orElse(null);

}

@Override

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Override

@Transactional

public void updateCountry(Country country) {

countryRepository.save(country);

}

@Override

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

@Override

public List<Country> searchCountries(String keyword) {

return countryRepository.findByNameContaining(keyword);

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) {

System.out.println("Start");

List<Country> allCountries = countryService.getAllCountries();

allCountries.forEach(System.out::println);

System.out.println(countryService.findCountryByCode("IN"));

Country newCountry = new Country("ZZ", "Zootopia");

countryService.addCountry(newCountry);

System.out.println("After adding Zootopia: " + countryService.findCountryByCode("ZZ"));

newCountry.setName("Zebra Zone");

countryService.updateCountry(newCountry);

System.out.println("After update: " + countryService.findCountryByCode("ZZ"));

countryService.deleteCountry("ZZ");

System.out.println("After delete: " + countryService.findCountryByCode("ZZ"));

List<Country> results = countryService.searchCountries("in");

System.out.println("Search results:");

results.forEach(System.out::println);

System.out.println("End");

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

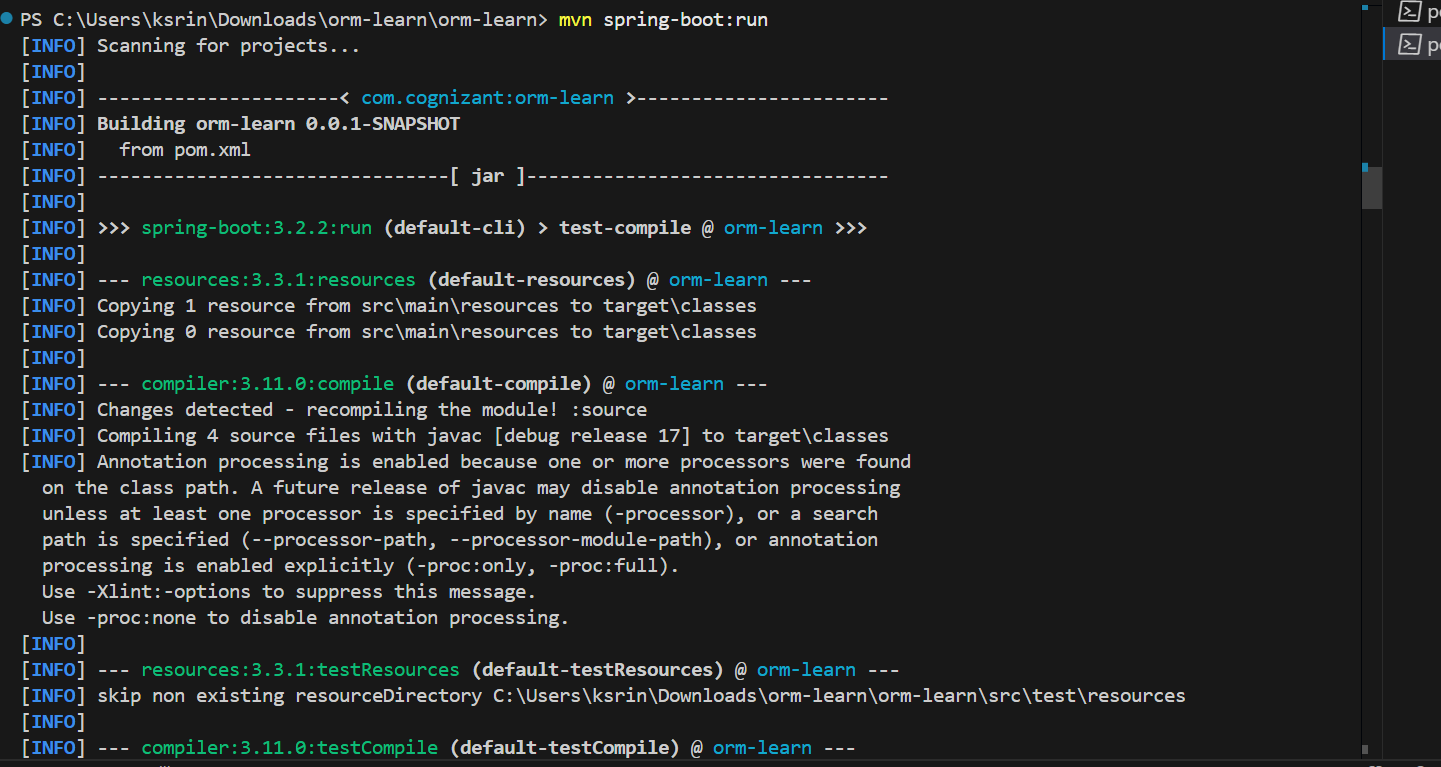
spring.datasource.password=root

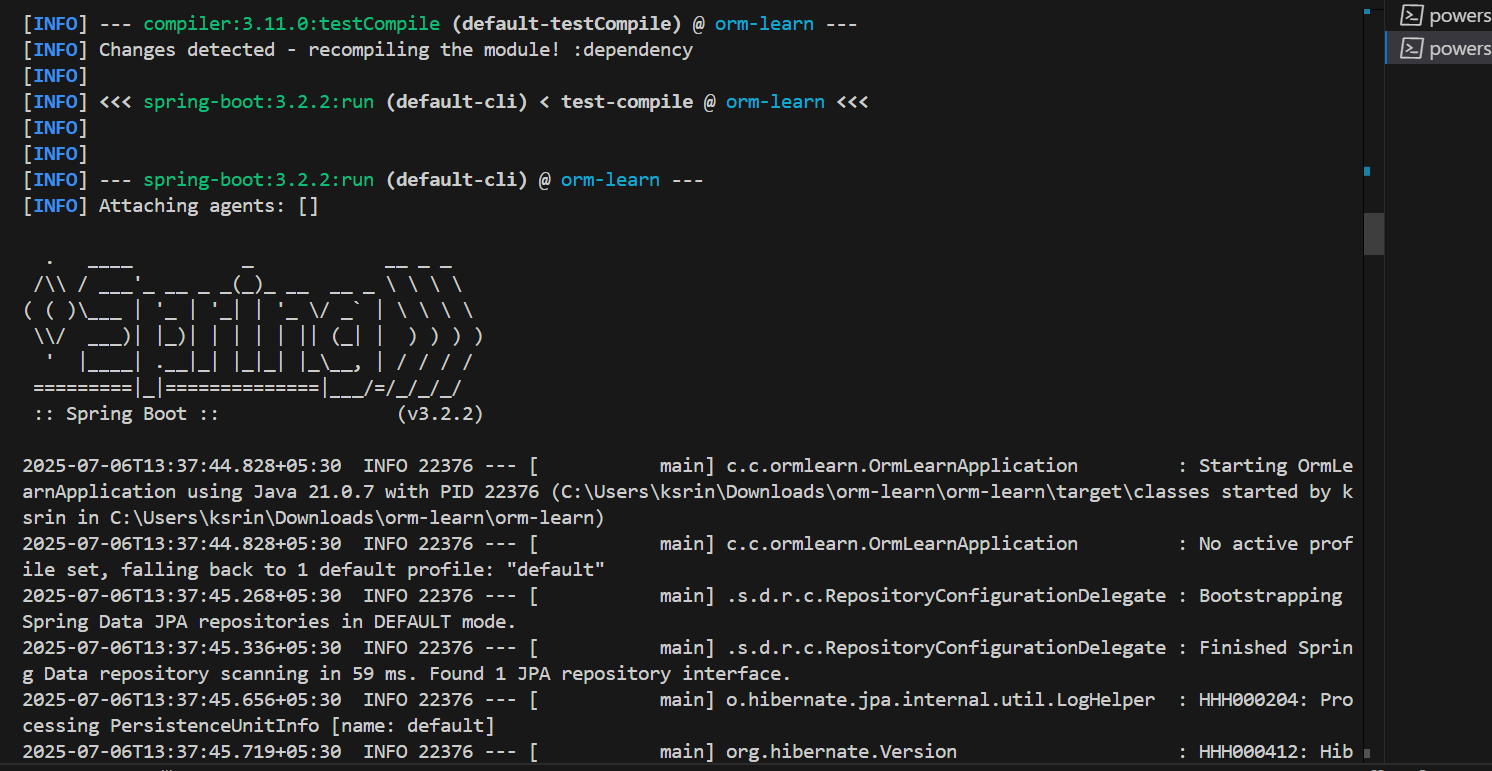
spring.jpa.hibernate.ddl-auto=none

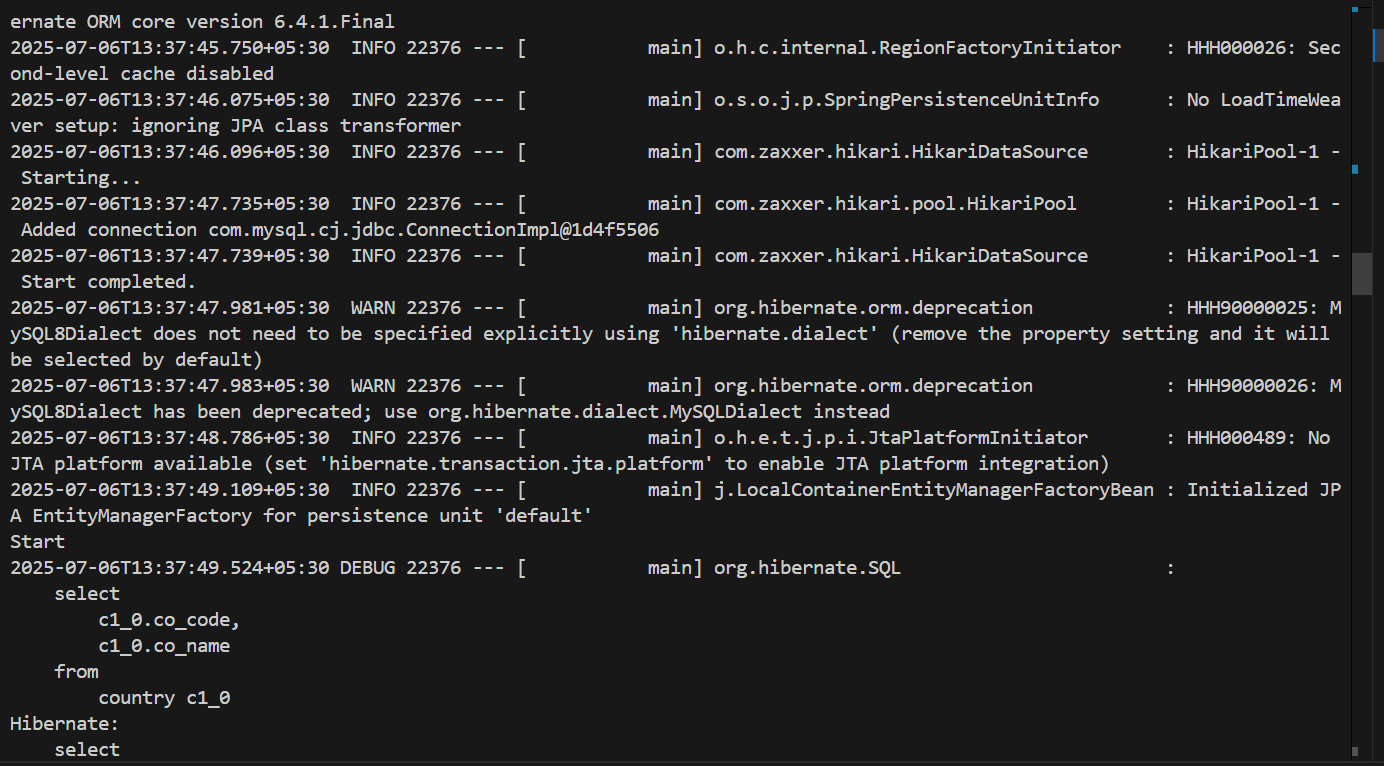
spring.jpa.show-sql=true

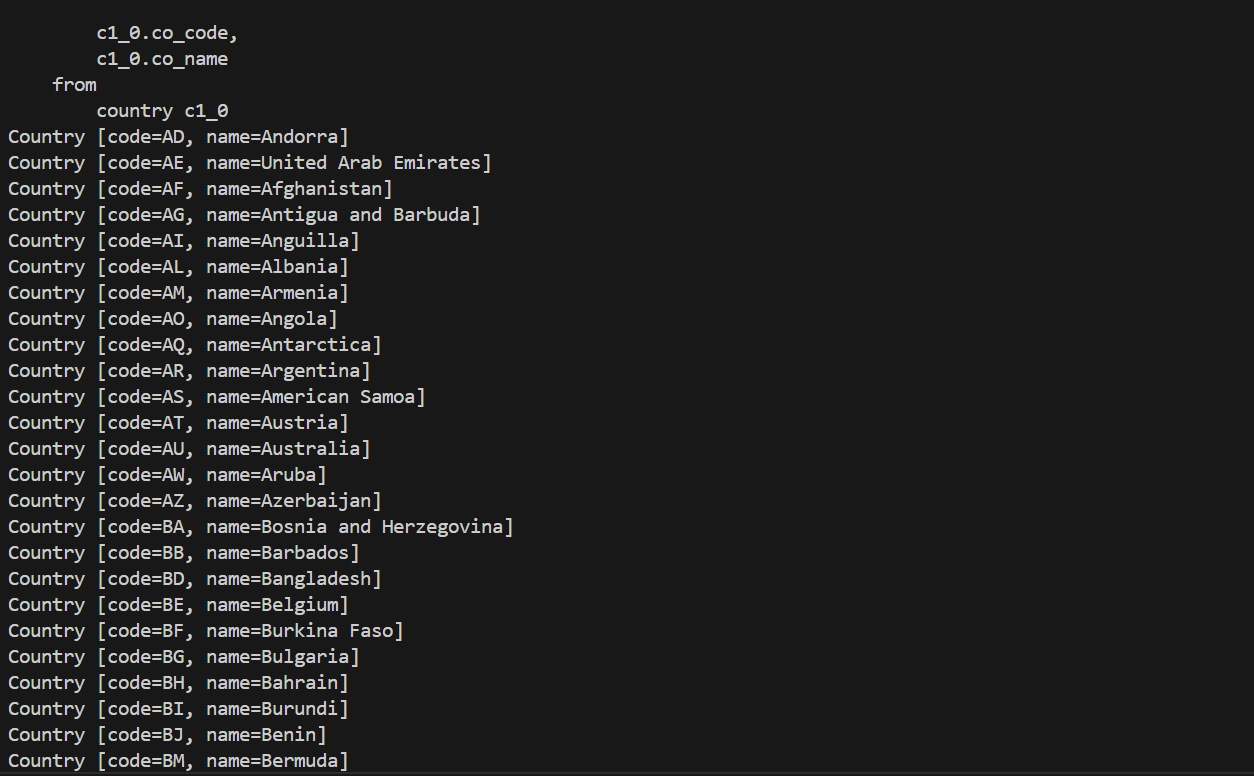
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

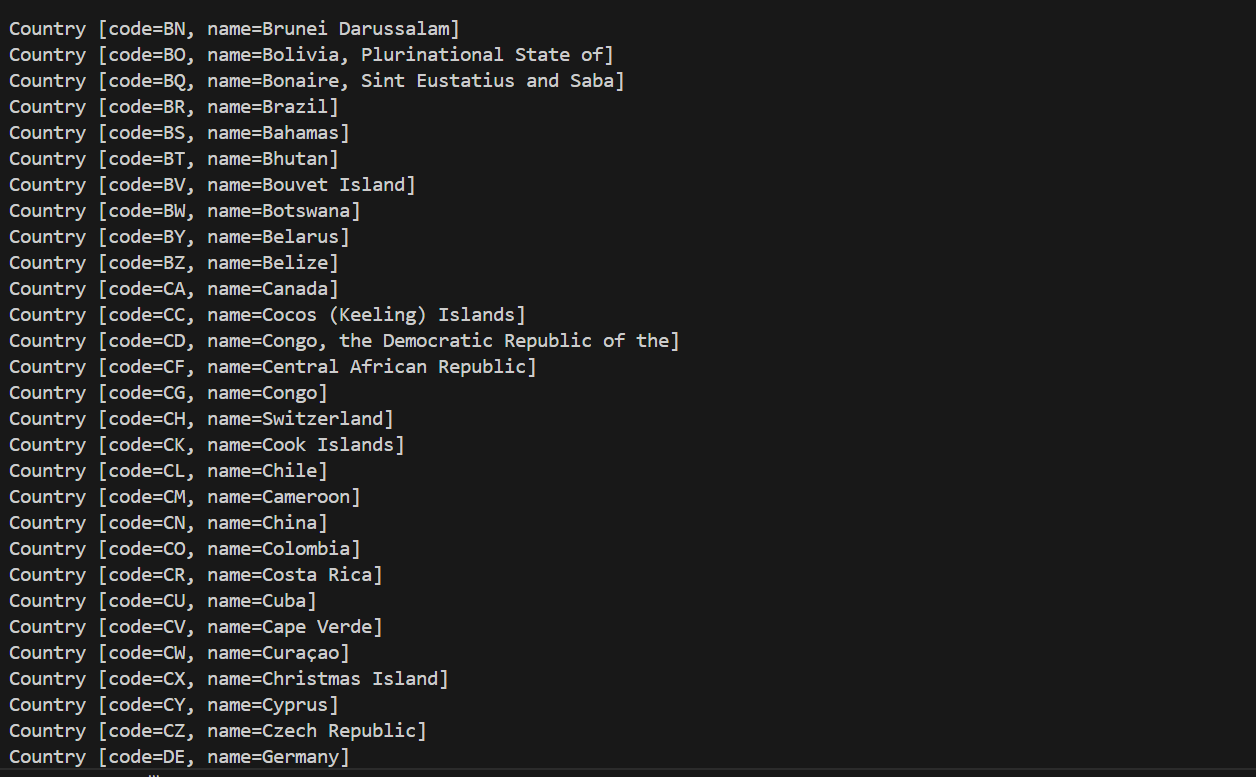
**Output:**

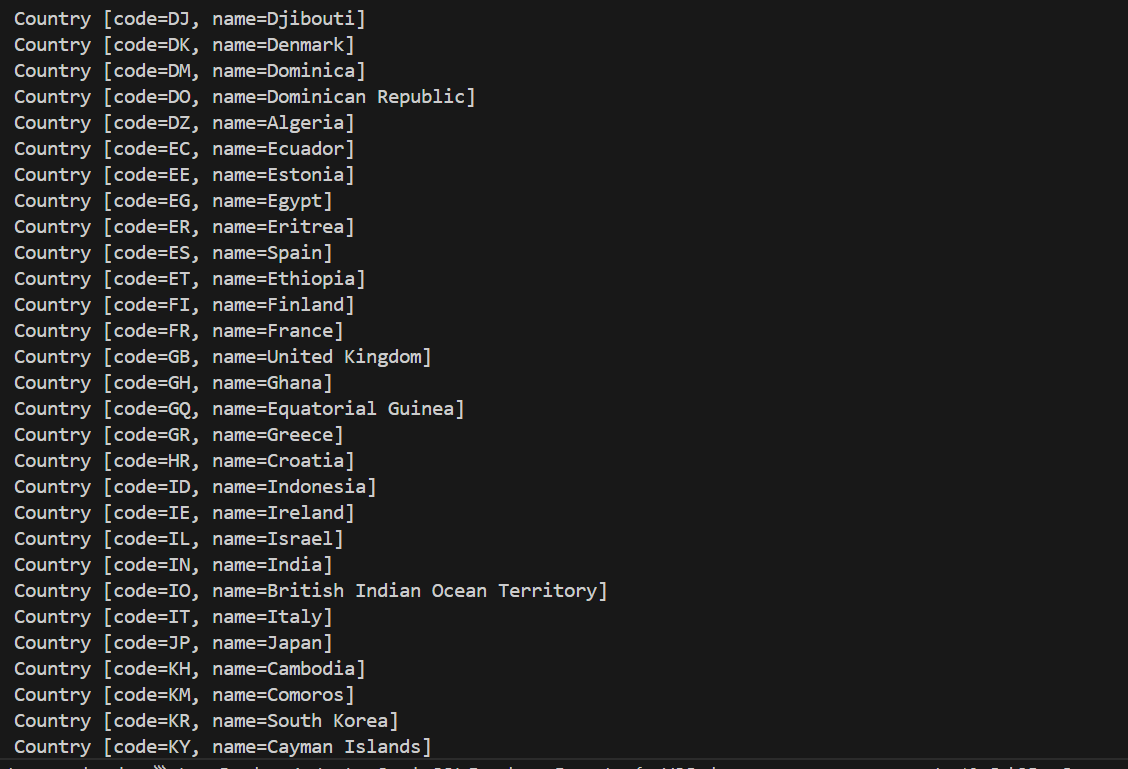


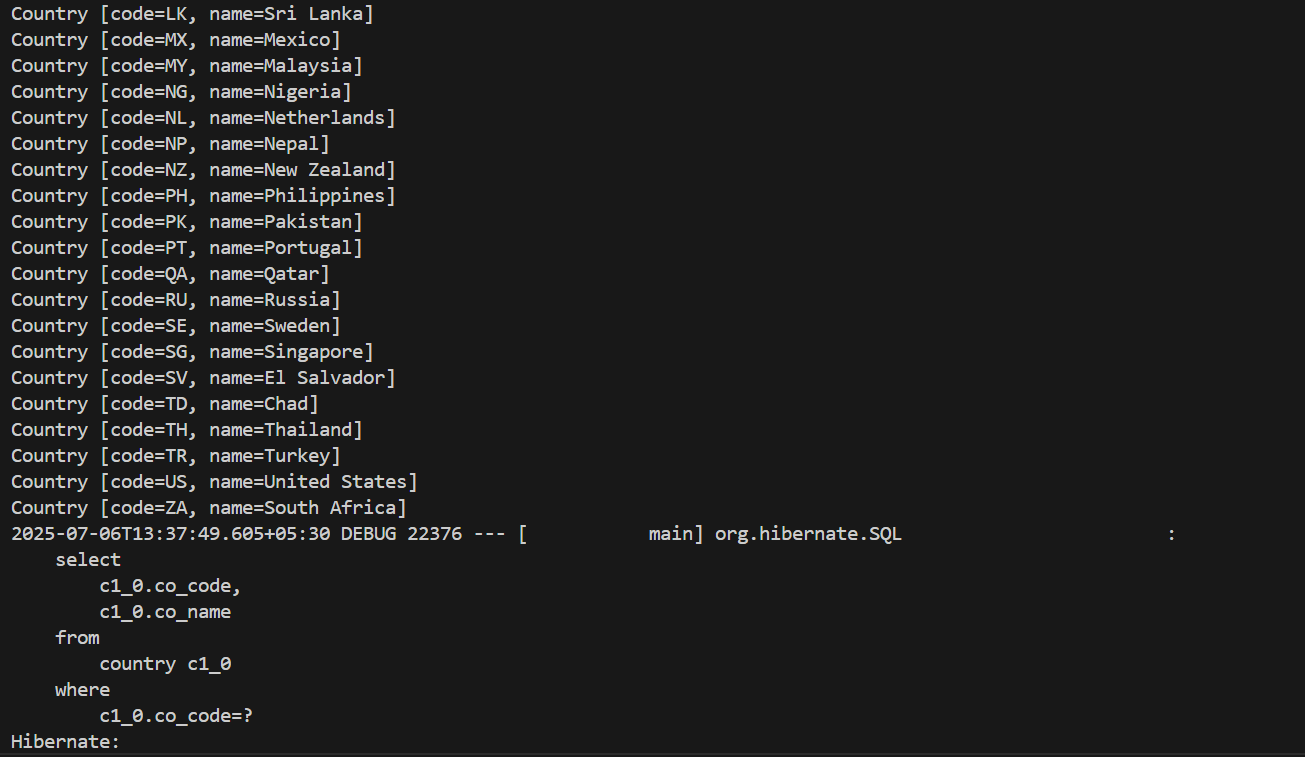


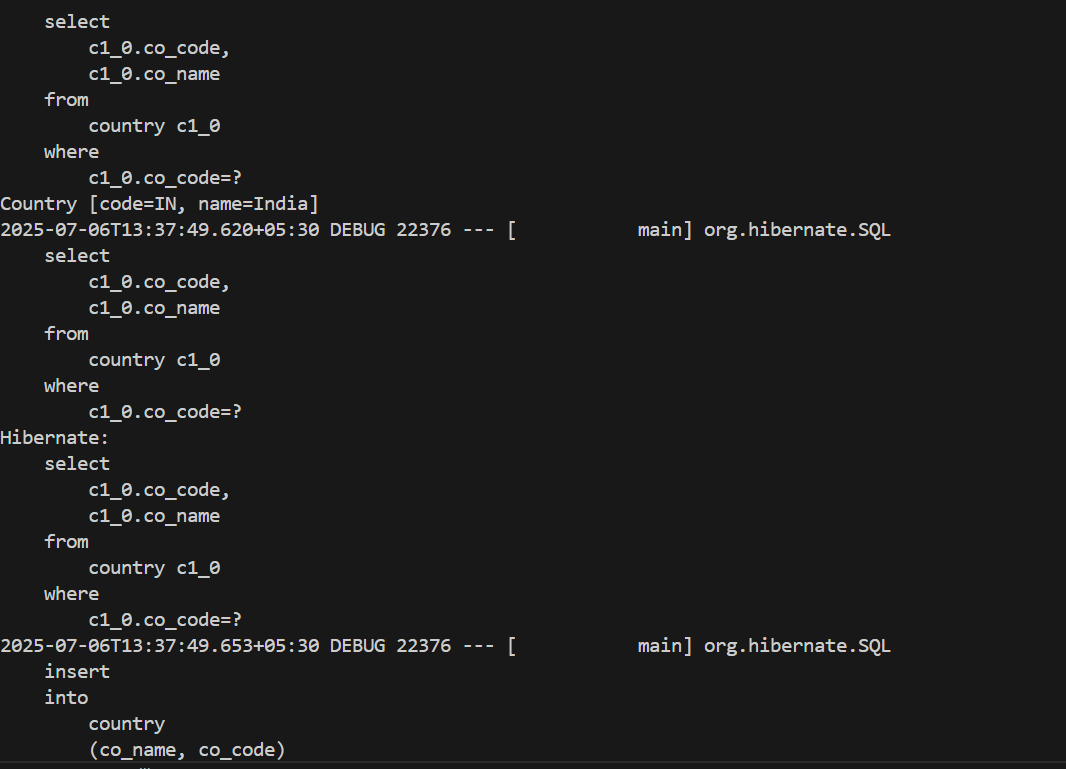


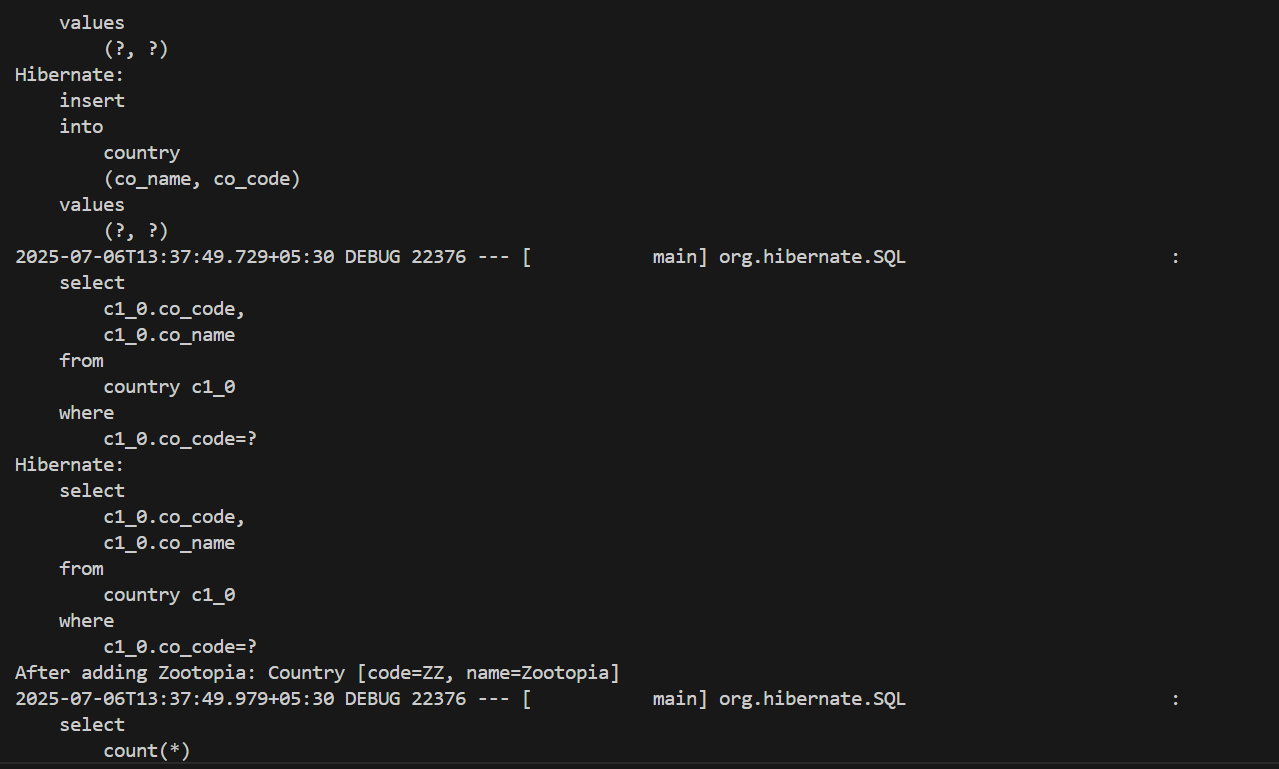


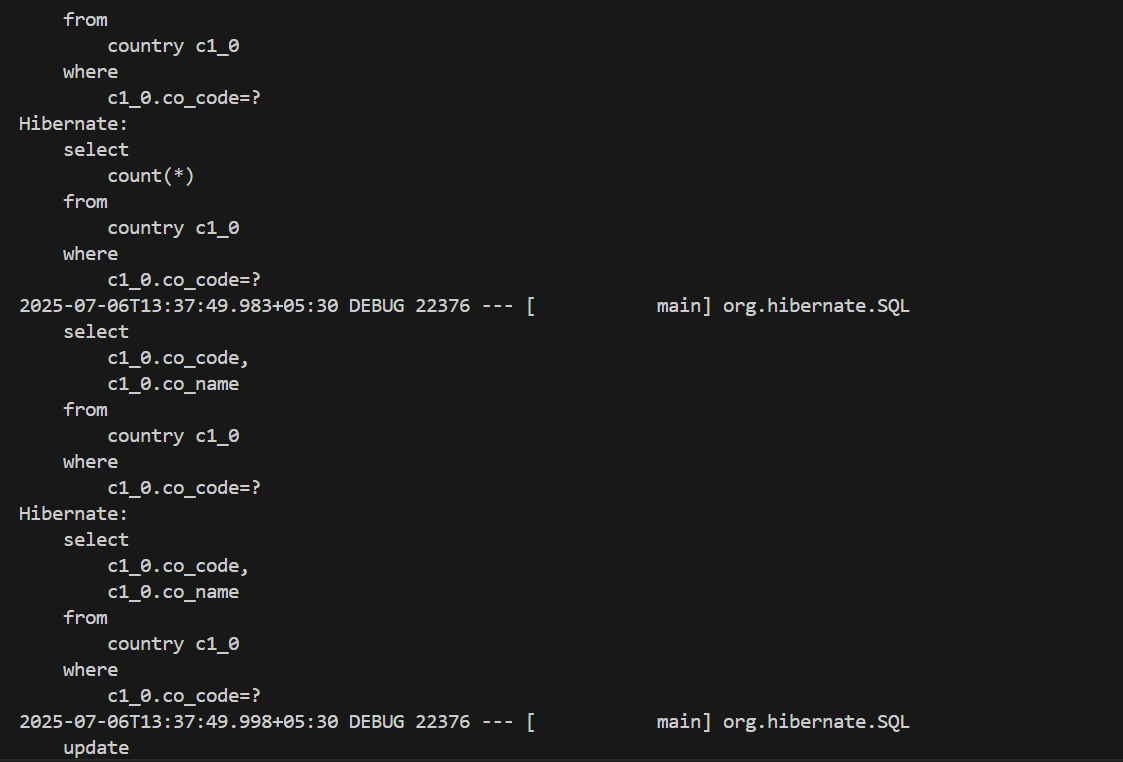


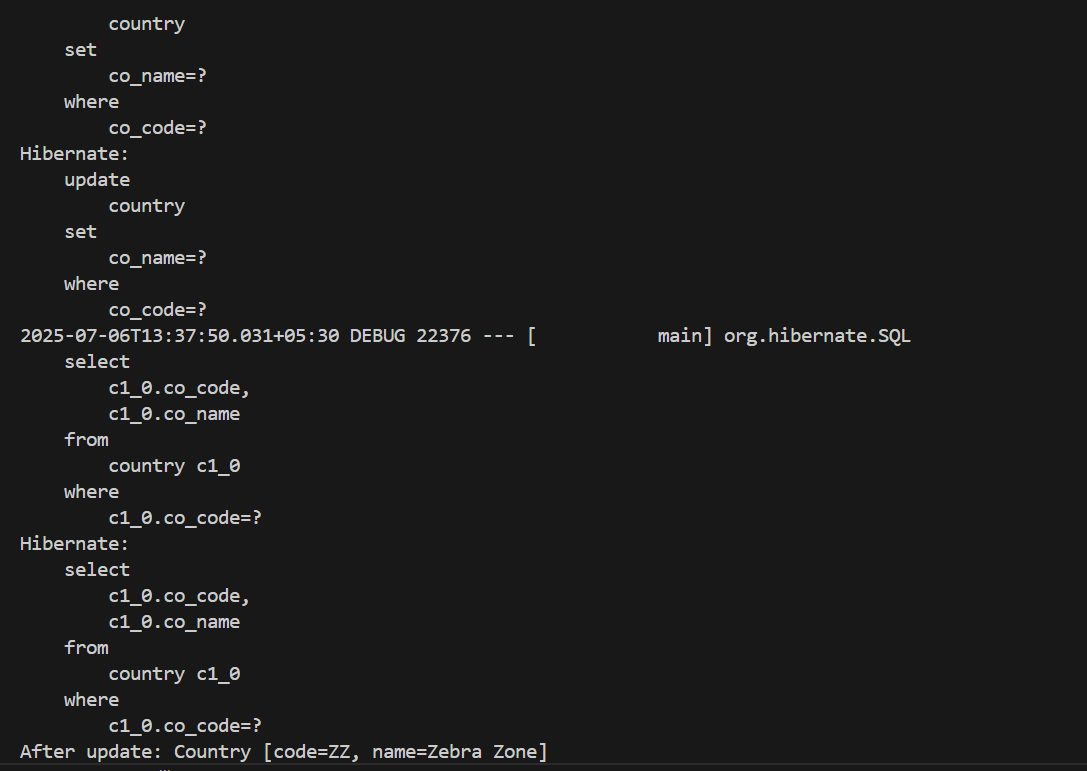


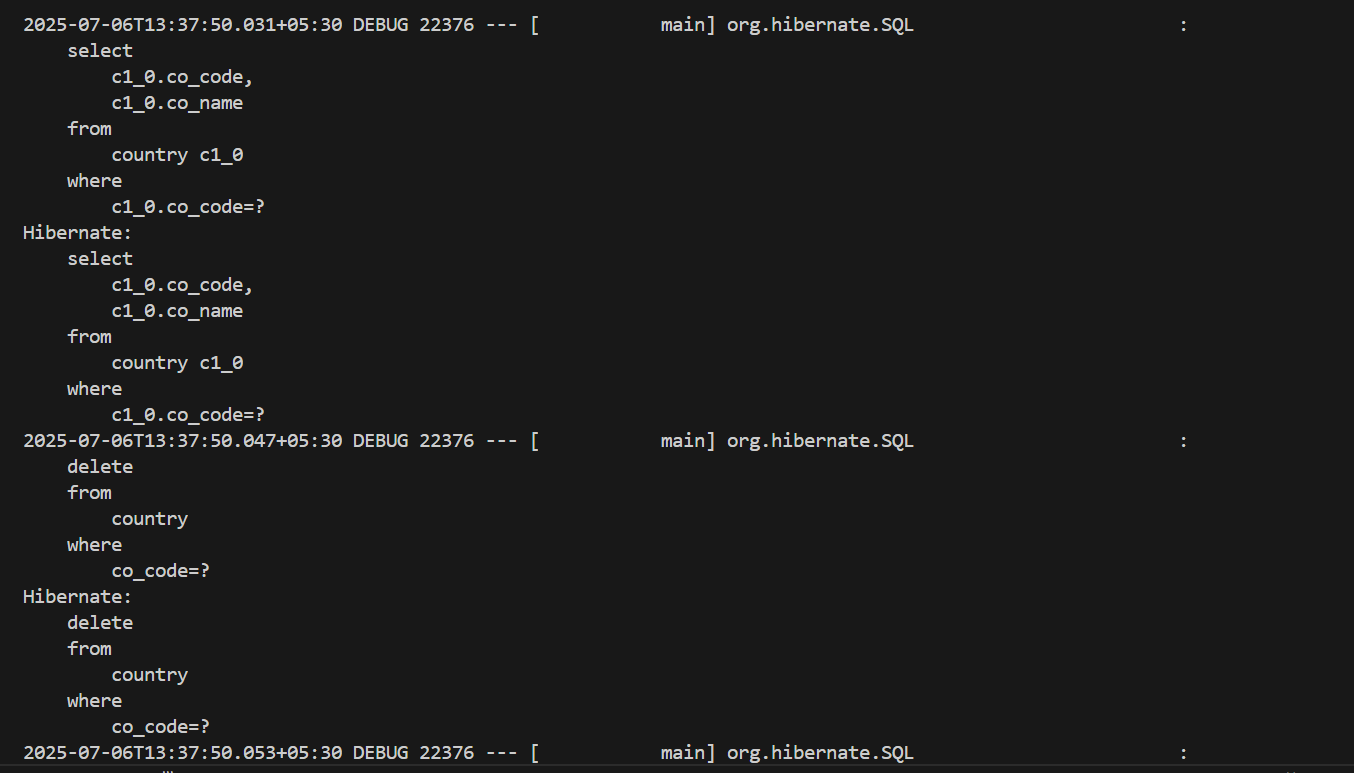


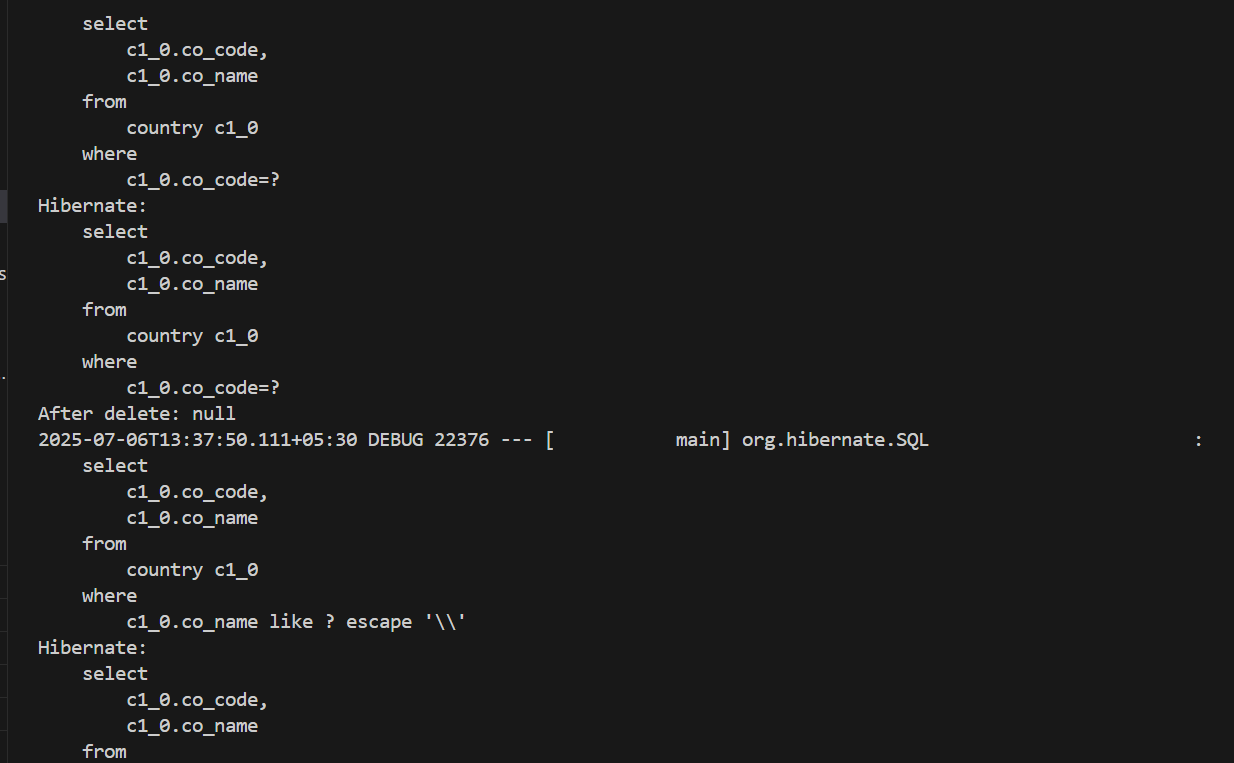


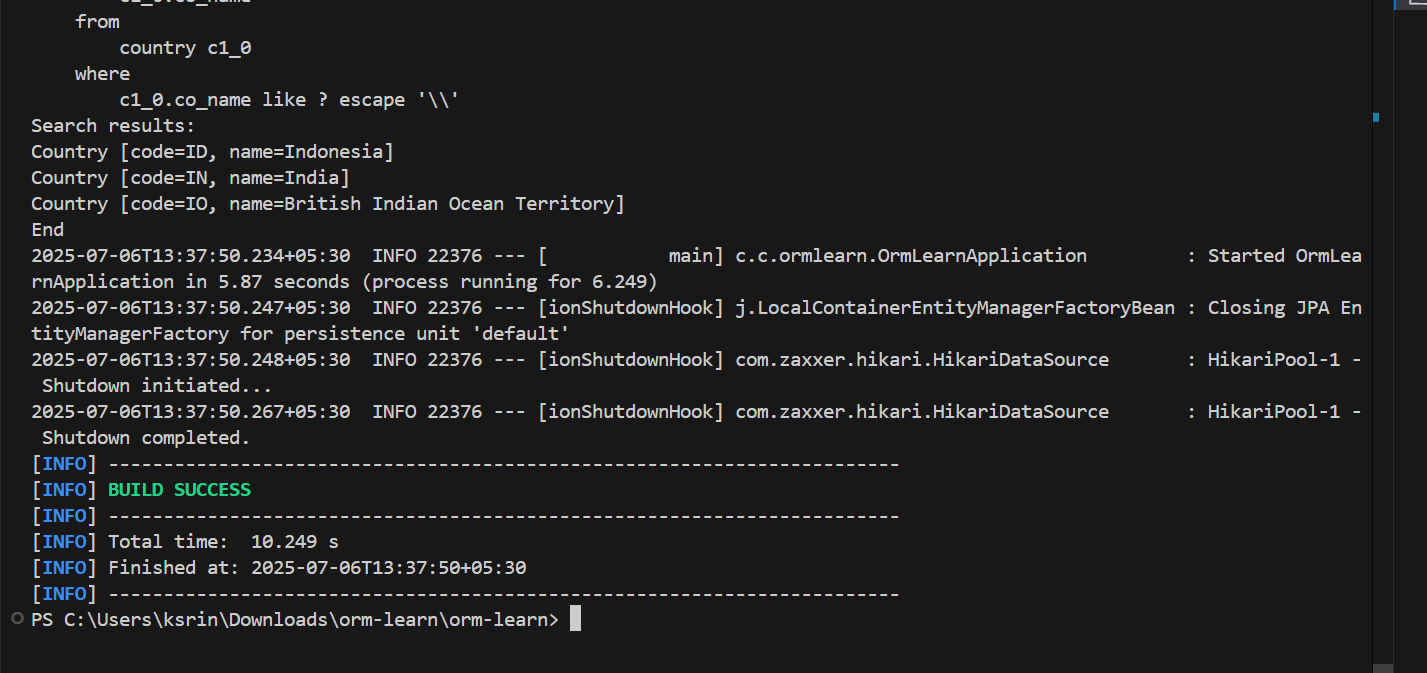












**Hands on 6**

**Find a country based on country code**

**Country.java**

package com.cognizant.ormlearn;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@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 "Country [code=" + code + ", name=" + name + "]";

}

}

**CountryRepository.java**

package com.cognizant.ormlearn;

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

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryNotFoundException.java**

package com.cognizant.ormlearn;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**CountryService.java**

package com.cognizant.ormlearn;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@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()) {

throw new CountryNotFoundException("Country not found with code: " + code);

}

return result.get();

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import jakarta.annotation.PostConstruct;

@SpringBootApplication

public class OrmLearnApplication {

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

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@PostConstruct

public void testFindCountryByCode() {

LOGGER.info("Start");

try {

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End");

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

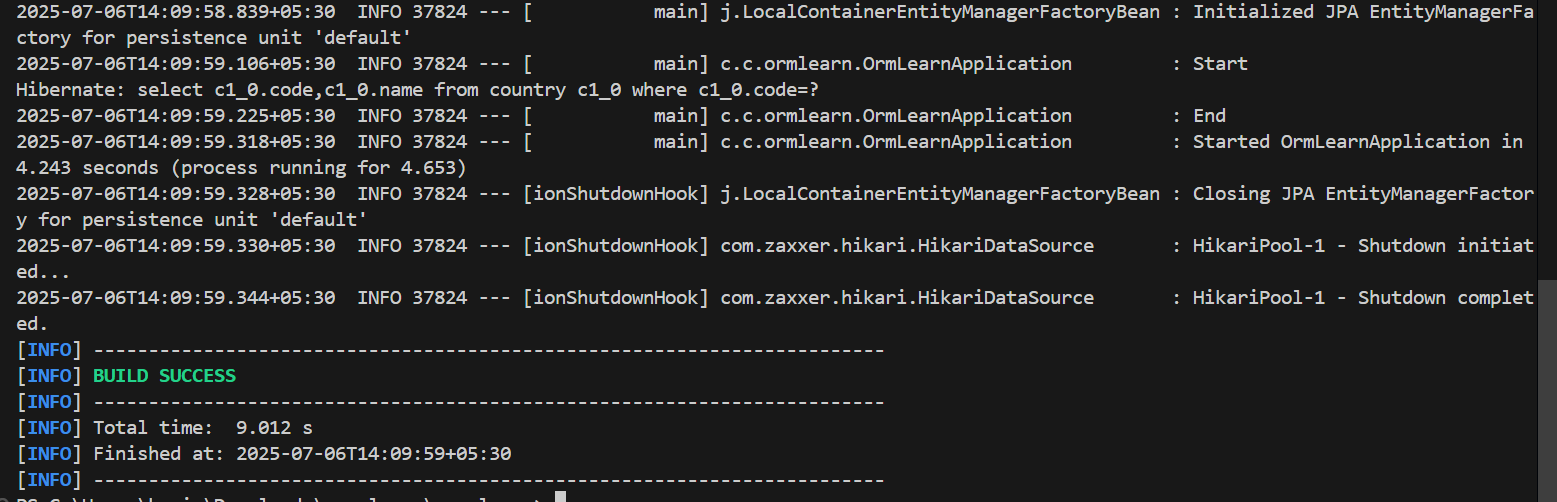
spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

logging.level.com.cognizant.ormlearn=DEBUG

**output:**



**Hands on 7**

**Add a new country** 

**Country.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = 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 com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.ormlearn.service;

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(readOnly = true)

public Country findCountryByCode(String code) {

return countryRepository.findById(code).orElse(null);

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.CommandLineRunner;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

testAddCountry();

}

public void testAddCountry() {

System.out.println("Start Add");

Country country = new Country("TS", "Testland");

countryService.addCountry(country);

System.out.println("End Add");

System.out.println("Start Find");

Country addedCountry = countryService.findCountryByCode("TS");

System.out.println("Added Country: " + addedCountry);

System.out.println("End Find");

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

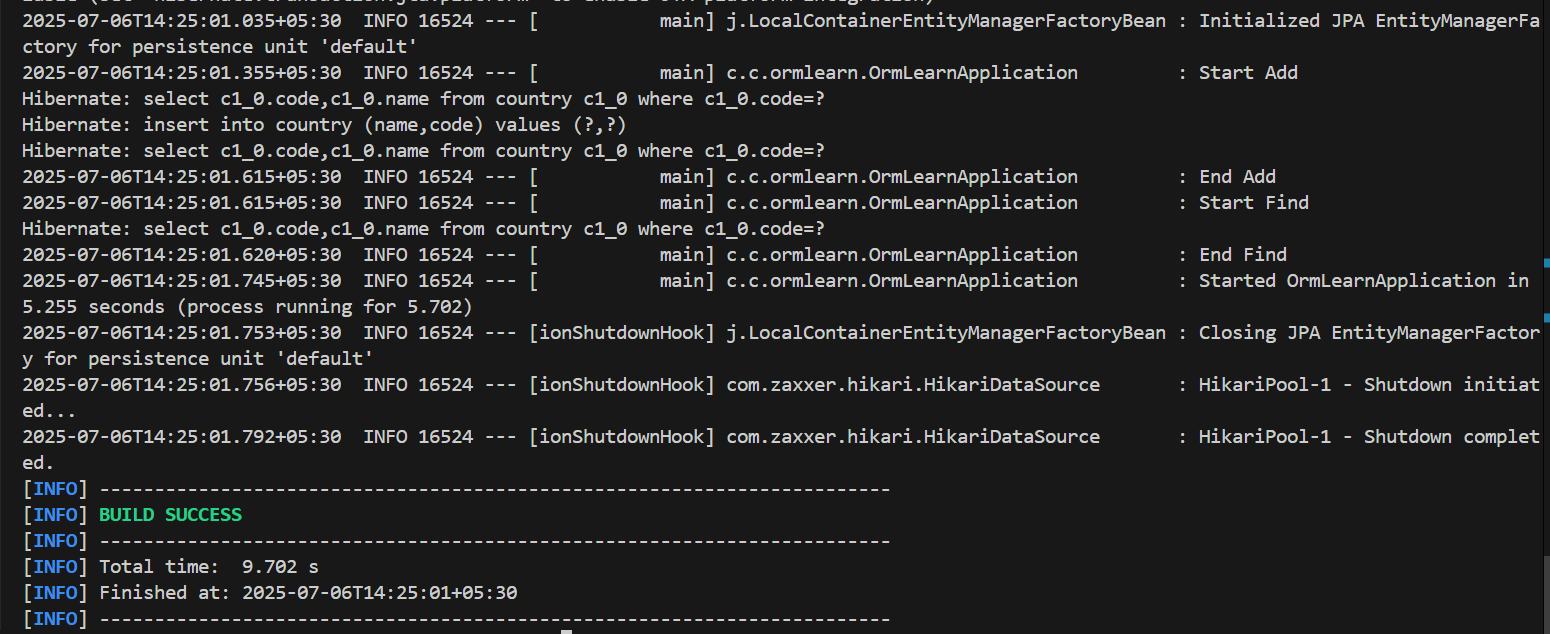
spring.datasource.username=root

spring.datasource.password=root

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

**output:**



**Hands on 8**

**Update a country based on code**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.orm\_learn.service.CountryService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

System.out.println("Start Update");

countryService.updateCountry("IN", "Bharat");

System.out.println("Updated Country: " + countryService.findCountryByCode("IN").getName());

System.out.println("End Update");

}

}

**Country.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@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;

}

}

**CountryRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Country;

public interface CountryService {

public Country findCountryByCode(String code);

public void updateCountry(String code, String name);

}

**CountryNotFoundException.java**

package com.cognizant.orm\_learn.exception;

public class CountryNotFoundException extends RuntimeException {

public CountryNotFoundException(String message) {

super(message);

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

spring.datasource.password=root

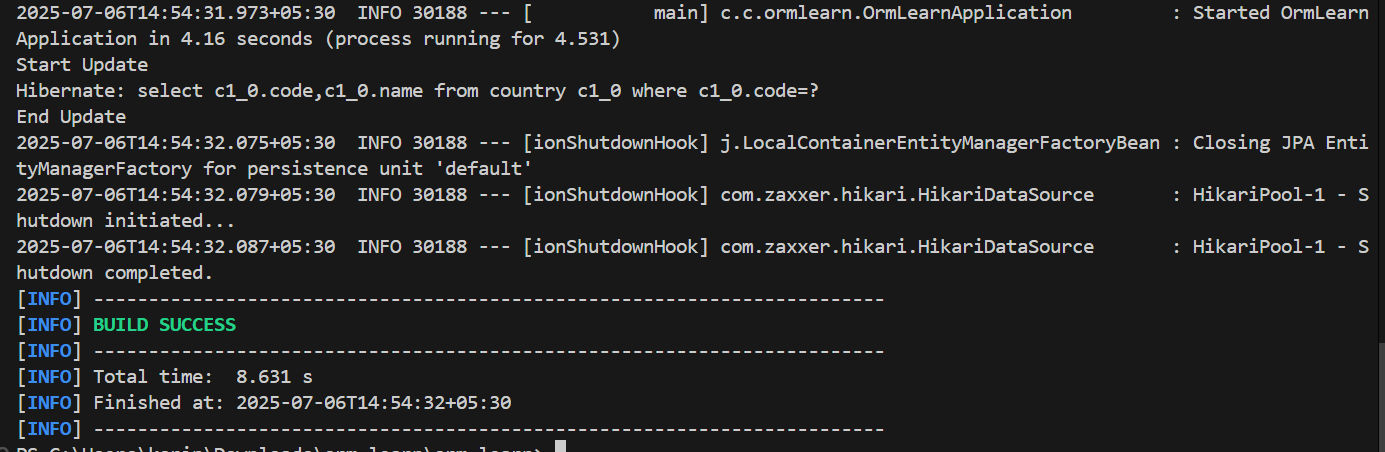
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**output:**



**Hands on 9**

**Delete a country based on code**

**Country.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@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 "Country [code=" + code + ", name=" + name + "]";

}

}

**CountryRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.orm\_learn.service;

import java.util.List;

import com.cognizant.orm\_learn.model.Country;

public interface CountryService {

List<Country> getAllCountries();

Country findCountryByCode(String code);

void updateCountry(String code, String name);

void deleteCountry(String code);

}

**CountryServiceImpl.java**

package com.cognizant.orm\_learn.service;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

import jakarta.transaction.Transactional;

@Service

public class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository countryRepository;

@Override

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

@Override

public Country findCountryByCode(String code) {

Optional<Country> result = countryRepository.findById(code);

return result.orElse(null);

}

@Override

@Transactional

public void updateCountry(String code, String name) {

Country country = findCountryByCode(code);

if (country != null) {

country.setName(name);

countryRepository.save(country);

}

}

@Override

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication {

public static void main(String[] args) {

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

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

System.out.println("Start Delete");

countryService.deleteCountry("IN");

Country deleted = countryService.findCountryByCode("IN");

if (deleted == null) {

System.out.println("Delete success: Country removed.");

} else {

System.out.println("Delete failed: Country still exists.");

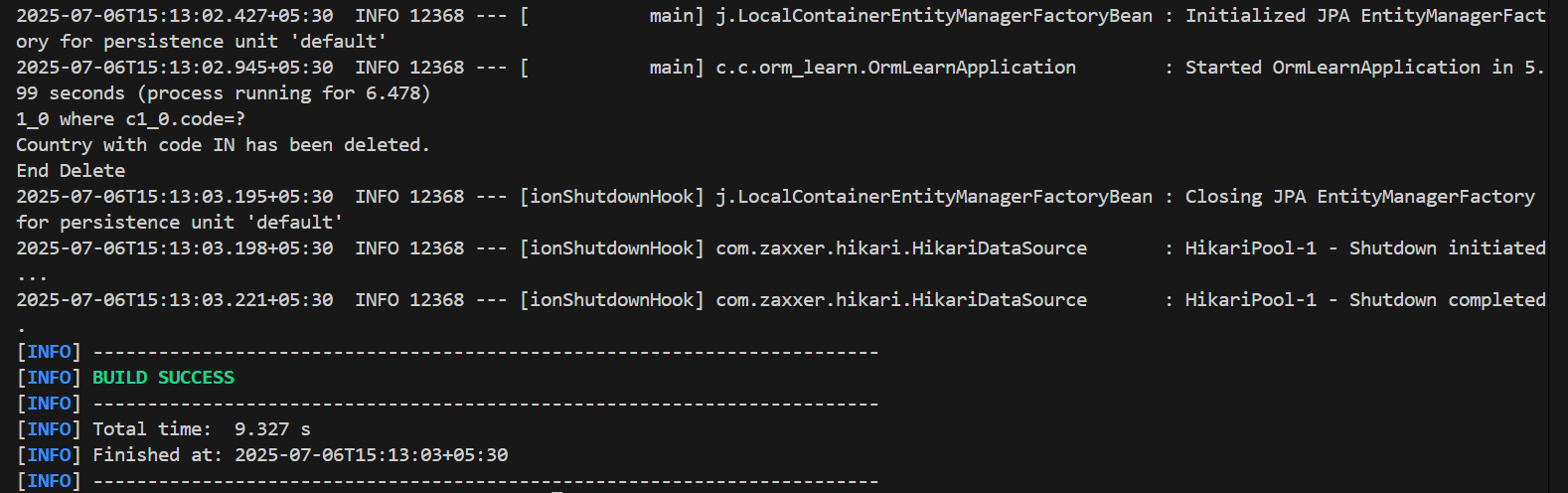
}

System.out.println("End Delete");

}

}

**Output:**



**Hands on 1**

**Write queries on country table using Query Methods**

**Country.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@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;

}

}

**CountryRepository.java**

package com.cognizant.orm\_learn;

import java.util.List;

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

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String keyword);

List<Country> findByNameContainingOrderByNameAsc(String keyword);

List<Country> findByNameStartingWith(String prefix);

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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) {

testFindByNameContaining();

testFindByNameContainingOrderByNameAsc();

testFindByNameStartingWith();

}

public void testFindByNameContaining() {

System.out.println("Searching for countries containing 'ou':");

List<Country> countries = countryRepository.findByNameContaining("ou");

for (Country country : countries) {

System.out.println(country.getCode() + " - " + country.getName());

}

}

public void testFindByNameContainingOrderByNameAsc() {

System.out.println("\nSearching for countries containing 'ou' (sorted):");

List<Country> countries = countryRepository.findByNameContainingOrderByNameAsc("ou");

for (Country country : countries) {

System.out.println(country.getCode() + " - " + country.getName());

}

}

public void testFindByNameStartingWith() {

System.out.println("\nSearching for countries starting with 'Z':");

List<Country> countries = countryRepository.findByNameStartingWith("Z");

for (Country country : countries) {

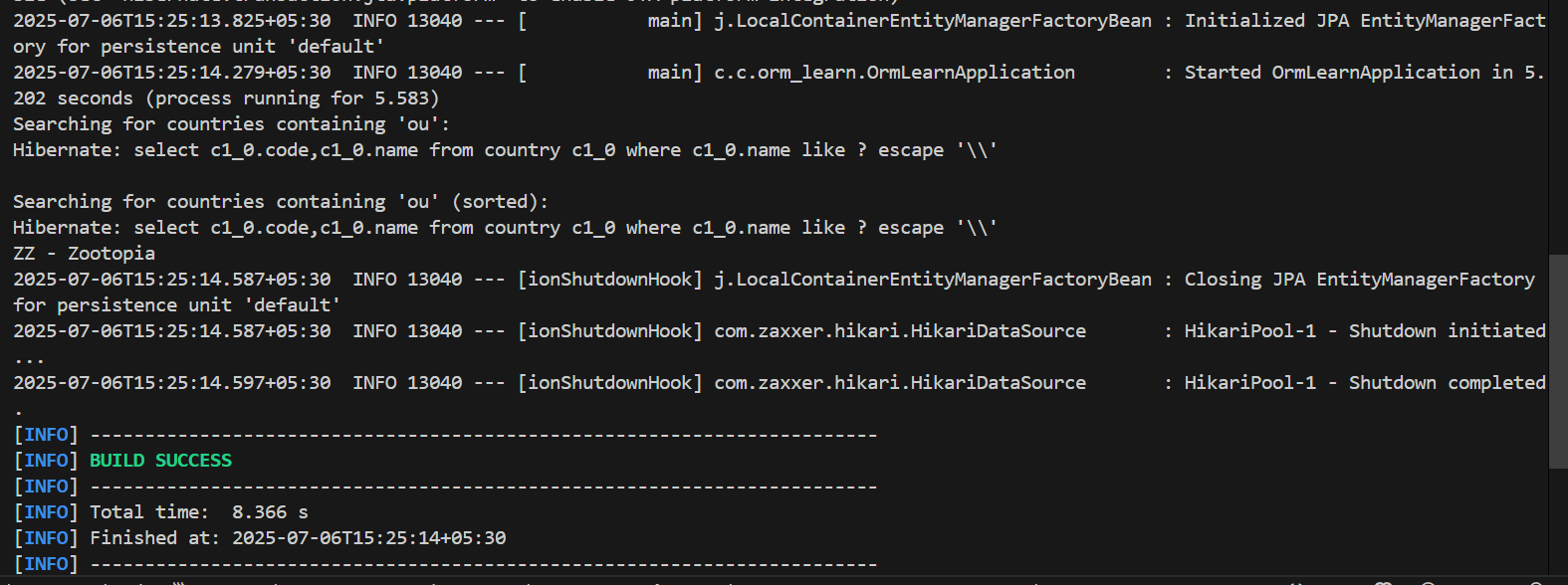
System.out.println(country.getCode() + " - " + country.getName());

}

}

}

**Output:**



**Hands on 2**

**Write queries on stock table using Query Methods**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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) {

try {

System.out.println("1. Facebook stocks in Sep 2019:");

List<Stock> fbStocks = stockRepository.findByStCodeAndStDateBetween(

"FB", LocalDate.of(2019, 9, 1), LocalDate.of(2019, 9, 30));

fbStocks.forEach(System.out::println);

System.out.println("\n2. Google stocks with close > 1250:");

stockRepository.findByStCodeAndStCloseGreaterThan("GOOGL", new BigDecimal("1250"))

.forEach(System.out::println);

System.out.println("\n3. Top 3 by volume:");

stockRepository.findTop3ByOrderByStVolumeDesc()

.forEach(System.out::println);

System.out.println("\n4. Netflix lowest 3 closing prices:");

stockRepository.findTop3ByStCodeOrderByStCloseAsc("NFLX")

.forEach(System.out::println);

} catch (Exception e) {

System.err.println("\uD83D\uDCA5 Error: " + e.getMessage());

e.printStackTrace();

}

}

}

**Stock.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDate;

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int stId;

private String stCode;

private LocalDate stDate;

private BigDecimal stOpen;

private BigDecimal stClose;

private long stVolume;

public int getStId() { return stId; }

public void setStId(int stId) { this.stId = stId; }

public String getStCode() { return stCode; }

public void setStCode(String stCode) { this.stCode = stCode; }

public LocalDate getStDate() { return stDate; }

public void setStDate(LocalDate stDate) { this.stDate = stDate; }

public BigDecimal getStOpen() { return stOpen; }

public void setStOpen(BigDecimal stOpen) { this.stOpen = stOpen; }

public BigDecimal getStClose() { return stClose; }

public void setStClose(BigDecimal stClose) { this.stClose = stClose; }

public long getStVolume() { return stVolume; }

public void setStVolume(long stVolume) { this.stVolume = stVolume; }

@Override

public String toString() {

return stCode + " | " + stDate + " | Open: " + stOpen + " | Close: " + stClose + " | Volume: " + stVolume;

}

}

**StockRepository.java**

package com.cognizant.orm\_learn;

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> findByStCodeAndStDateBetween(String stCode, LocalDate startDate, LocalDate endDate);

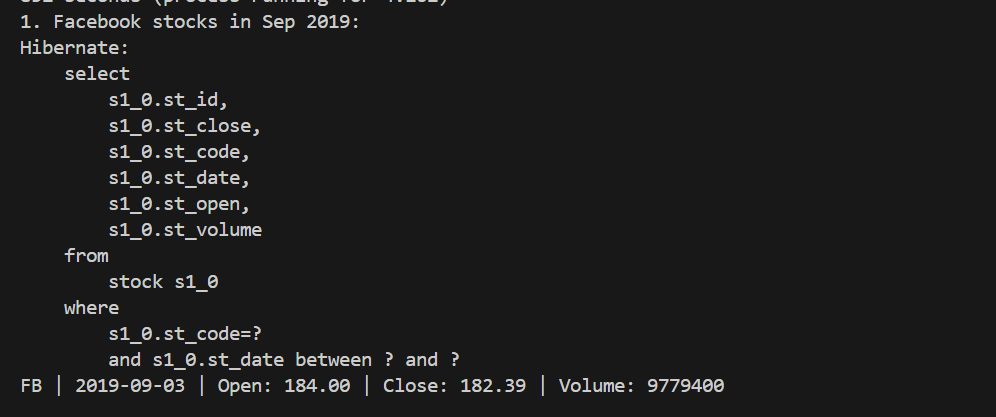
List<Stock> findByStCodeAndStCloseGreaterThan(String stCode, BigDecimal close);

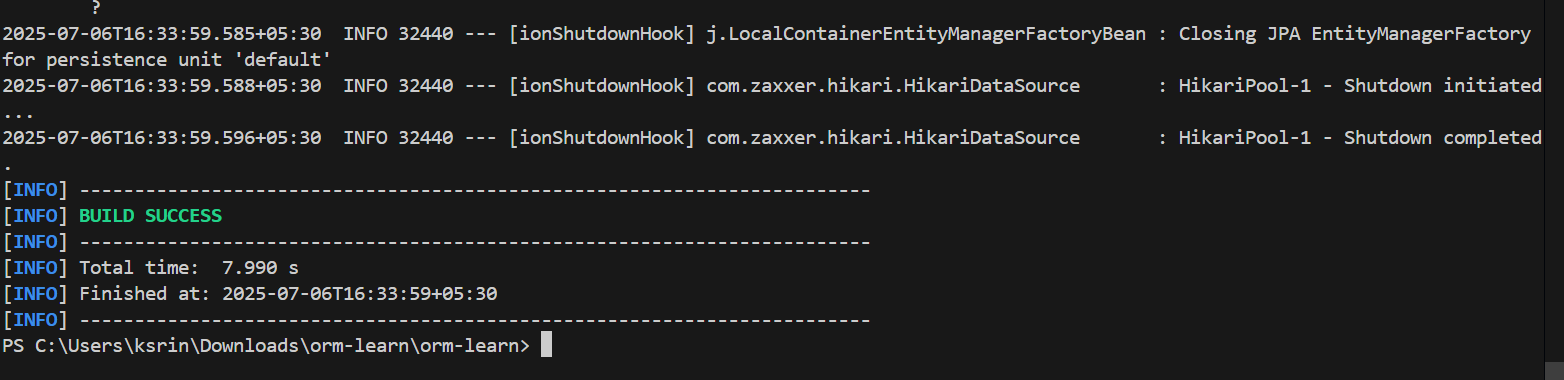
List<Stock> findTop3ByOrderByStVolumeDesc();

List<Stock> findTop3ByStCodeOrderByStCloseAsc(String stCode);

}

**Output:**





**Hands on 3**

**Create payroll tables and bean mapping**   
  
**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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;

import com.cognizant.orm\_learn.Employee;

import com.cognizant.orm\_learn.EmployeeRepository;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private EmployeeRepository employeeRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

System.out.println("All Employees:");

List<Employee> employees = employeeRepository.findAll();

employees.forEach(System.out::println);

}

}

**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@Column(name = "salary")

private double salary;

@Column(name = "permanent")

private boolean permanent;

@Column(name = "date\_of\_birth")

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(

name = "employee\_skill",

joinColumns = @JoinColumn(name = "employee\_id"),

inverseJoinColumns = @JoinColumn(name = "skill\_id")

)

private List<Skill> skillList;

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +

", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth +

", department=" + department + ", skills=" + skillList + "]";

}

}

**Department.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@Override

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

**Skill.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

@Entity

@Table(name = "skill")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@Override

public String toString() {

return "Skill [id=" + id + ", name=" + name + "]";

}

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**DepartmentRepository.java**

package com.cognizant.orm\_learn;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**SkillRepository.java**

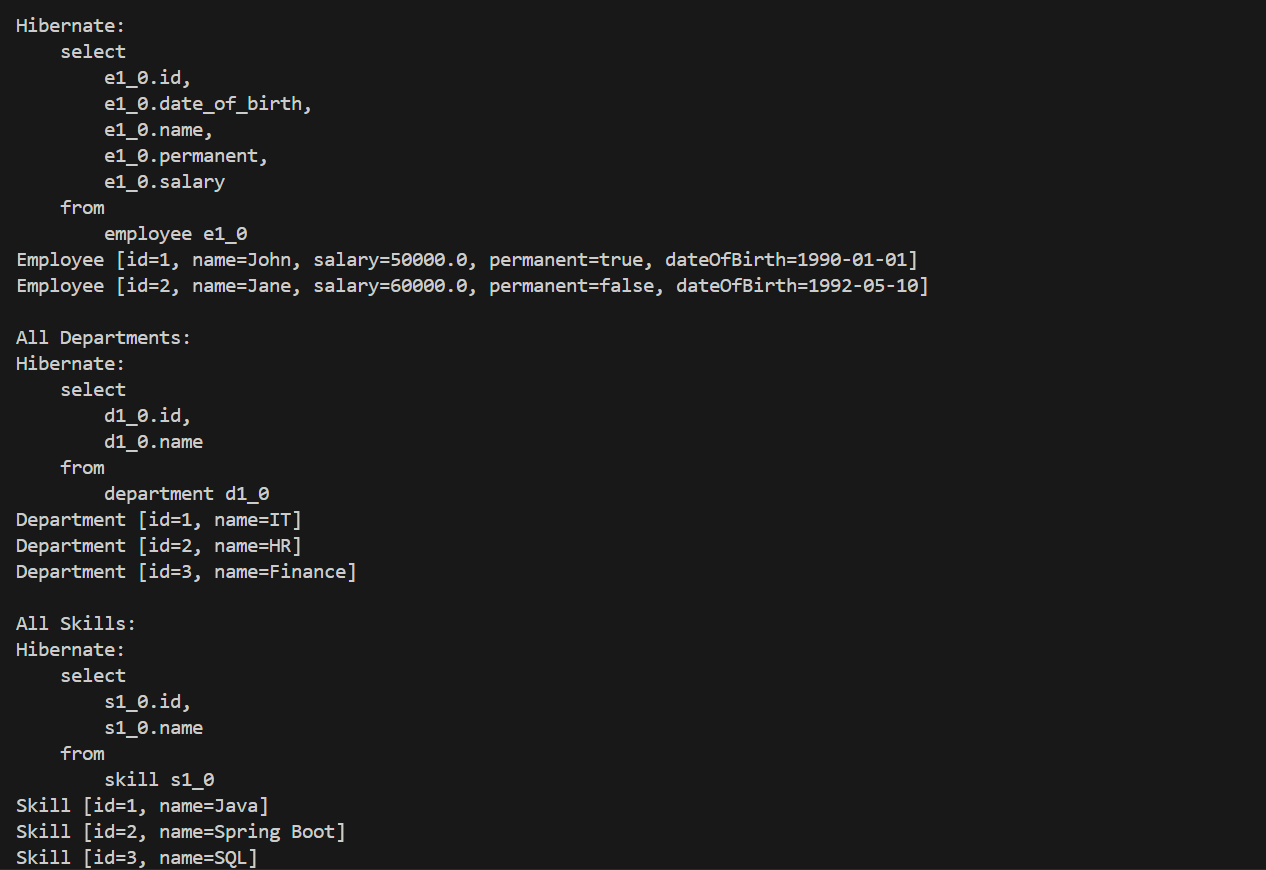
package com.cognizant.orm\_learn;

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

public interface SkillRepository extends JpaRepository<Skill, Integer> {

}

**Output:**



**Hands on 4**

**Implement many to one relationship between Employee and Department**   
  
**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.model.Employee;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

@Autowired

private DepartmentService departmentService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

testAddEmployee();

testGetEmployee();

testUpdateEmployee();

}

void testAddEmployee() throws Exception {

System.out.println("Start");

Employee employee = new Employee();

employee.setId(1);

employee.setName("John");

employee.setSalary(10000);

employee.setPermanent(true);

employee.setDateOfBirth(new SimpleDateFormat("dd/MM/yyyy").parse("01/01/2000"));

Department dept = departmentService.get(1);

employee.setDepartment(dept);

employeeService.save(employee);

System.out.println("End");

}

void testGetEmployee() {

System.out.println("Start");

Employee emp = employeeService.get(1);

System.out.println(emp);

System.out.println("End");

}

void testUpdateEmployee() throws Exception {

System.out.println("Start");

Employee employee = employeeService.get(1);

employee.setName("John Updated");

employee.setSalary(12000);

Department newDept = departmentService.get(1);

employee.setDepartment(newDept);

employeeService.save(employee);

System.out.println("End");

}

}

**Employee.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Date;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "em\_id")

private int id;

@Column(name = "em\_name")

private String name;

@Column(name = "em\_salary")

private double salary;

@Column(name = "em\_permanent")

private boolean permanent;

@Temporal(TemporalType.DATE)

@Column(name = "em\_date\_of\_birth")

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "em\_dp\_id")

private Department department;

}

**Department.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "dp\_id")

private int id;

@Column(name = "dp\_name")

private String name;

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**DepartmentRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**EmployeeService.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.model.Employee;

import com.cognizant.orm\_learn.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public void save(Employee emp) {

employeeRepository.save(emp);

}

public Employee get(int id) {

return employeeRepository.findById(id).orElse(null);

}

}

**DepartmentService.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.repository.DepartmentRepository;

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

public Department get(int id) {

return departmentRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Department with id " + id + " not found"));

}

}

**application.properties**

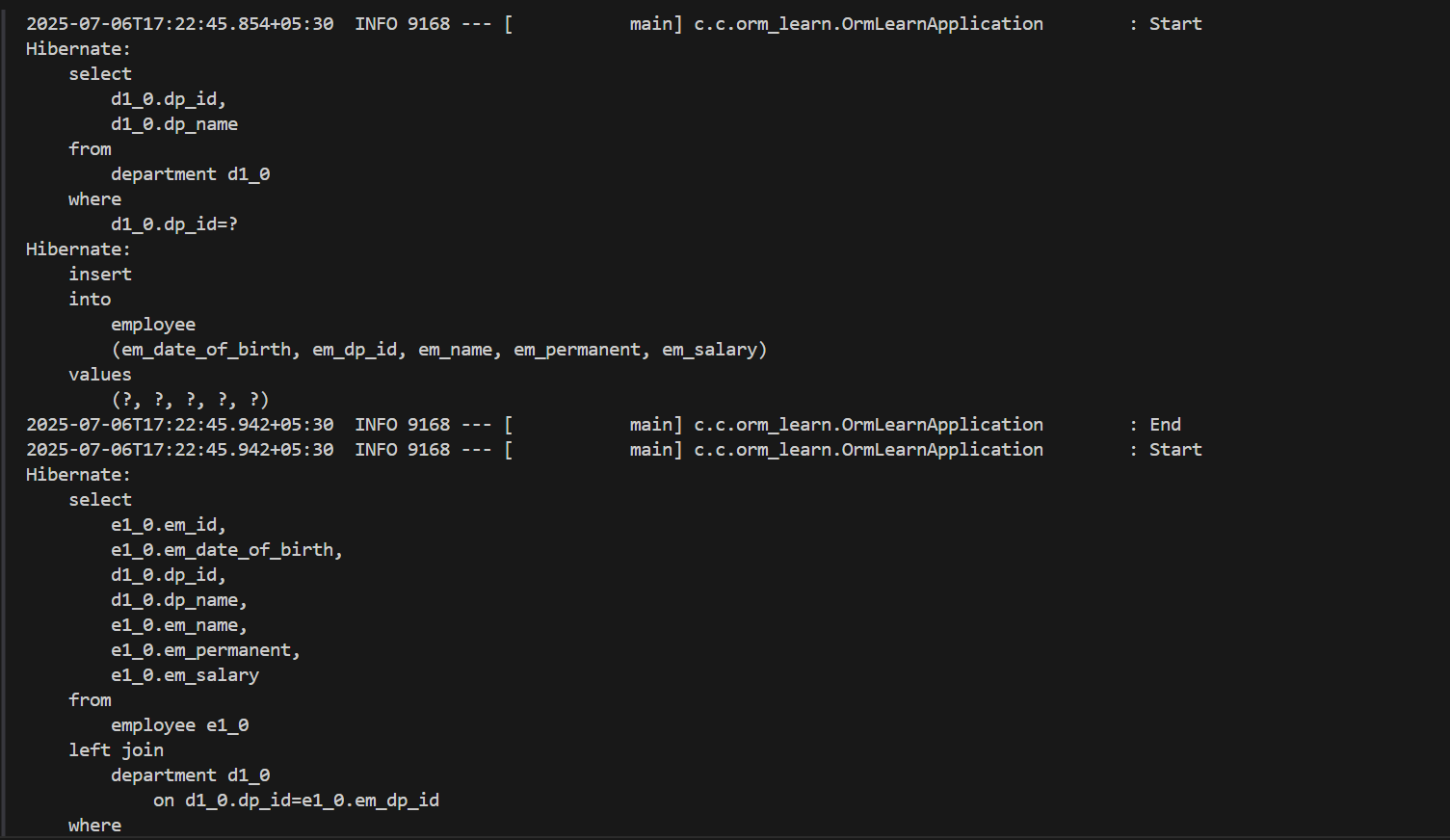
spring.datasource.url=jdbc:mysql://localhost:3306/your\_database\_name

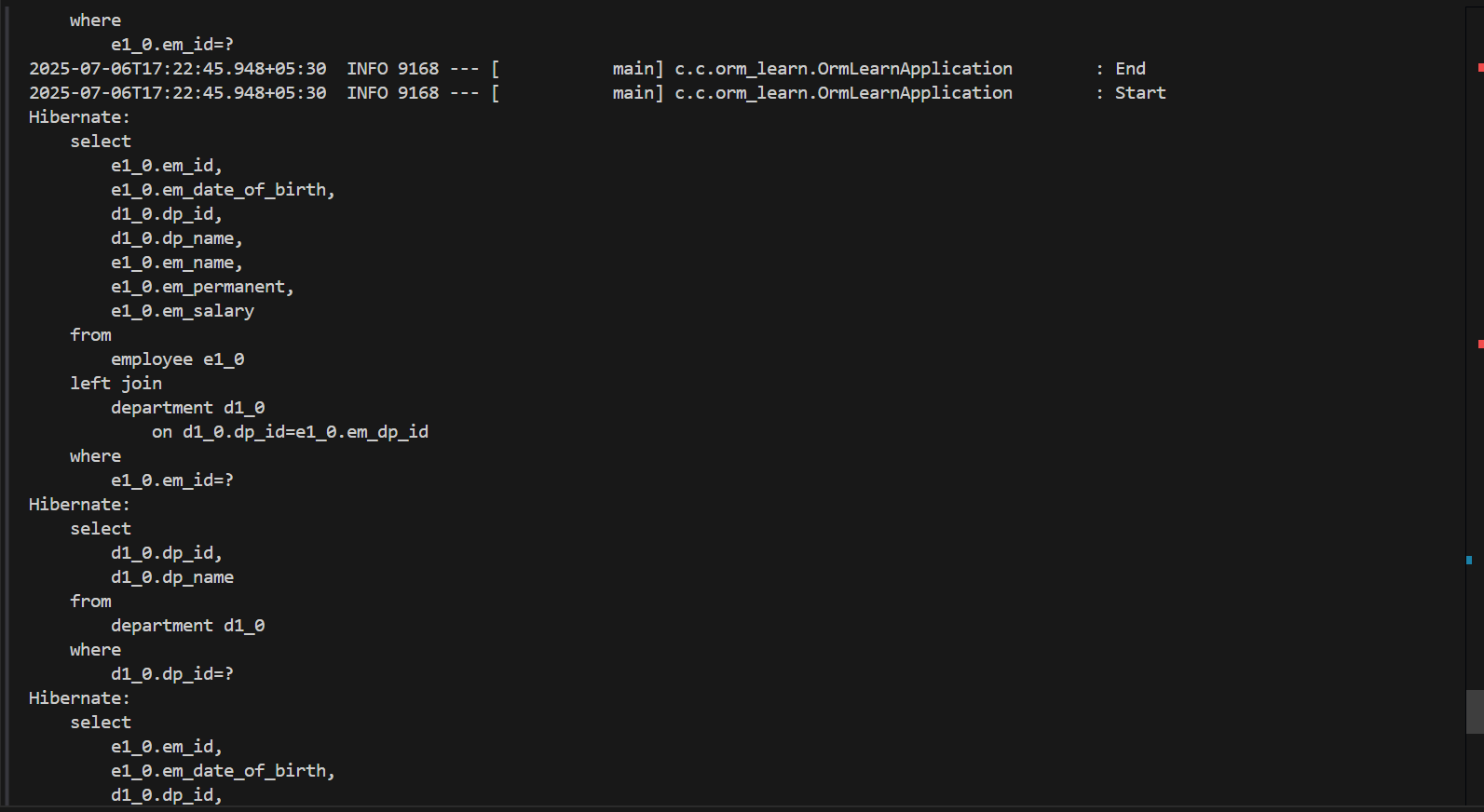
spring.datasource.username=root

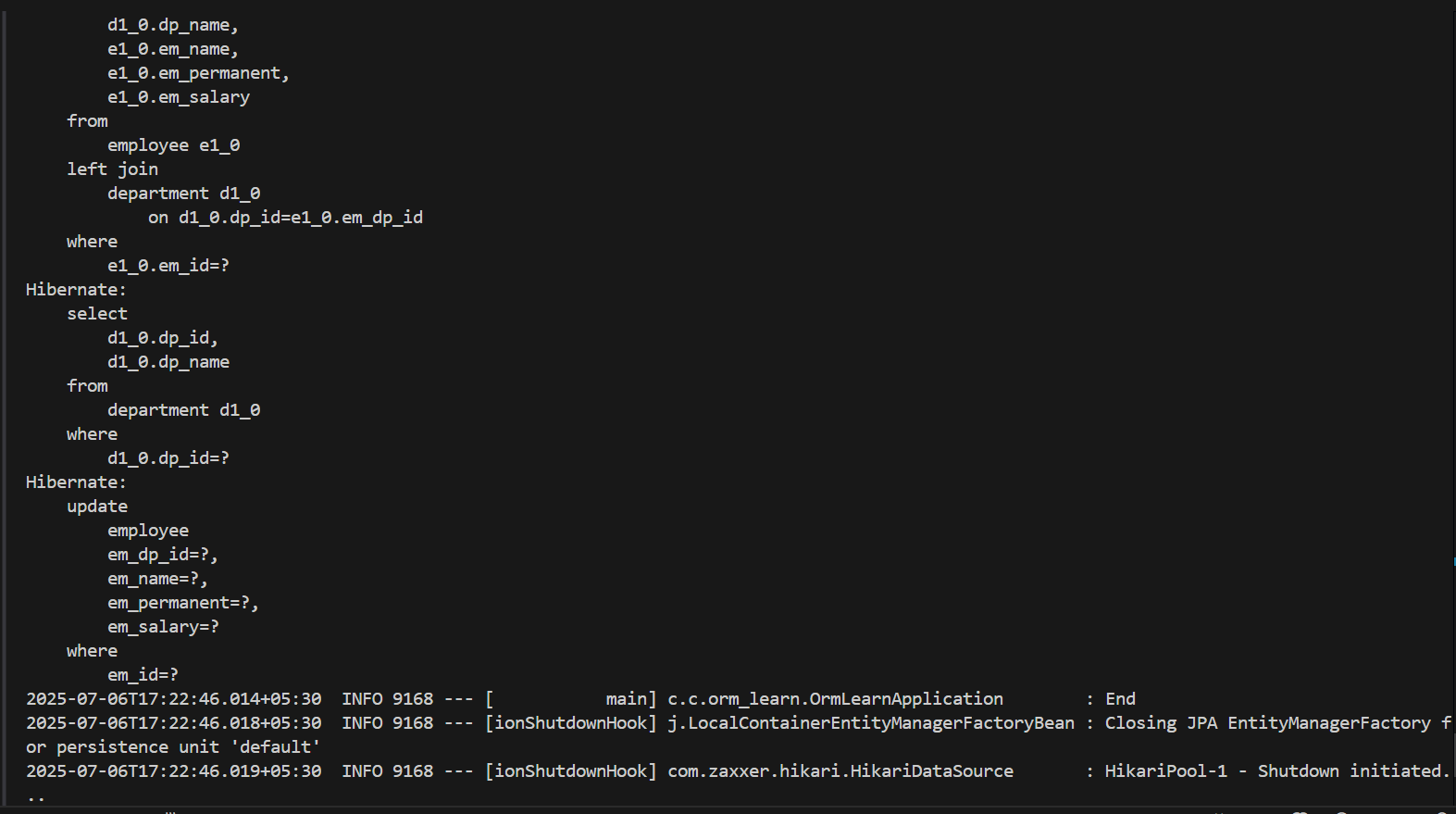
spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**output:**





**Hands on 5**

**Implement one to many relationship between Employee and Department**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import java.util.Set;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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 {

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

@Autowired

private DepartmentService departmentService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

void testGetDepartment() {

LOGGER.info("Start");

Department department = departmentService.get(1); // ensure dept ID 1 has multiple employees

LOGGER.debug("Department: {}", department);

Set<Employee> employeeList = department.getEmployeeList();

LOGGER.debug("Employees: {}", employeeList);

LOGGER.info("End");

}

}

**Department.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@OneToMany(mappedBy = "department", fetch = FetchType.EAGER)

private Set<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 Set<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(Set<Employee> employeeList) {

this.employeeList = employeeList;

}

@Override

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.Date;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private double salary;

private boolean permanent;

@Temporal(TemporalType.DATE)

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "em\_dp\_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 Date getDateOfBirth() {

return dateOfBirth;

}

public void setDateOfBirth(Date dateOfBirth) {

this.dateOfBirth = dateOfBirth;

}

public Department getDepartment() {

return department;

}

public void setDepartment(Department department) {

this.department = department;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";

}

}

**DepartmentRepository.java**

package com.cognizant.orm\_learn;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**DepartmentService.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

public Department get(int id) {

return departmentRepository.findById(id).orElseThrow(() -> new RuntimeException("Department with id " + id + " not found"));

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/your\_db\_name

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**SQL:**

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

salary DECIMAL(10, 2),

permanent BOOLEAN,

date\_of\_birth DATE,

em\_dp\_id INT,

FOREIGN KEY (em\_dp\_id) REFERENCES department(id)

);

-- Insert into Department

INSERT INTO department (id, name) VALUES (1, 'Technology');

-- Insert into Employee (linked to Department with ID 1)

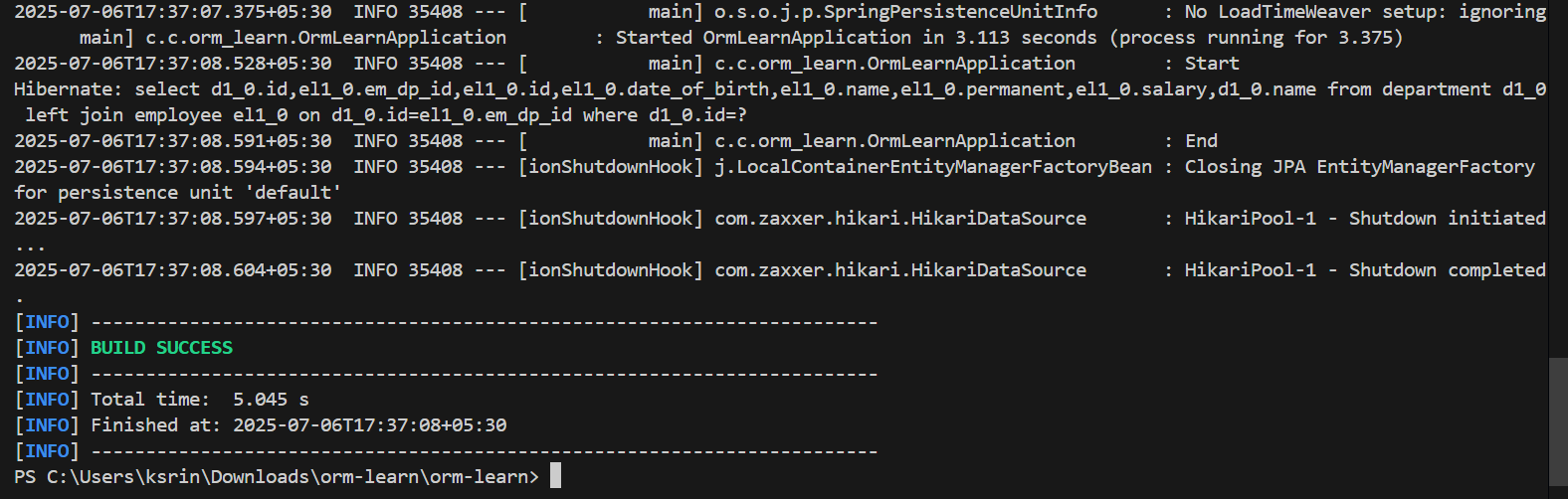
INSERT INTO employee (id, name, salary, permanent, date\_of\_birth, em\_dp\_id) VALUES

(101, 'John Doe', 50000.00, TRUE, '1990-05-15', 1),

(102, 'Jane Smith', 60000.00, TRUE, '1988-08-25', 1),

(103, 'Alice Johnson', 55000.00, FALSE, '1992-03-12', 1);

**output:**



**Hands on 6**

**Implement many to many relationship between Employee and Skill**   
  
**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.\*;

@Entity

public class Employee {

@Id

private int id;

private String name;

private double salary;

private boolean permanent;

@Column(name = "date\_of\_birth")

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "em\_dp\_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 Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

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; }

}

**Skill.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.\*;

@Entity

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; }

}

**SkillRepository.java**

package com.cognizant.orm\_learn;

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

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

**SkillService.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class SkillService {

@Autowired

private SkillRepository skillRepository;

public Skill get(int id) {

return skillRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Skill with id " + id + " not found"));

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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.Set;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private EmployeeService employeeService;

@Autowired

private SkillService skillService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

testAddSkillToEmployee();

}

private void testAddSkillToEmployee() {

LOGGER.info("Start");

Employee employee = employeeService.get(1);

Skill skill = skillService.get(2);

Set<Skill> skillList = employee.getSkillList();

skillList.add(skill);

employee.setSkillList(skillList);

employeeService.save(employee);

LOGGER.debug("Updated Skills: {}", employee.getSkillList());

LOGGER.info("End");

}

}

**SQL:**

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(50),

salary DOUBLE,

permanent BOOLEAN,

date\_of\_birth DATE,

em\_dp\_id INT

);

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE skill (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE employee\_skill (

es\_em\_id INT,

es\_sk\_id INT,

PRIMARY KEY (es\_em\_id, es\_sk\_id),

FOREIGN KEY (es\_em\_id) REFERENCES employee(id),

FOREIGN KEY (es\_sk\_id) REFERENCES skill(id)

);

-- Departments

INSERT INTO department (id, name) VALUES (1, 'IT'), (2, 'HR');

-- Employees

INSERT INTO employee (id, name, salary, permanent, date\_of\_birth, em\_dp\_id)

VALUES (1, 'John', 50000, true, '1990-01-01', 1),

(2, 'Jane', 60000, false, '1992-05-10', 1);

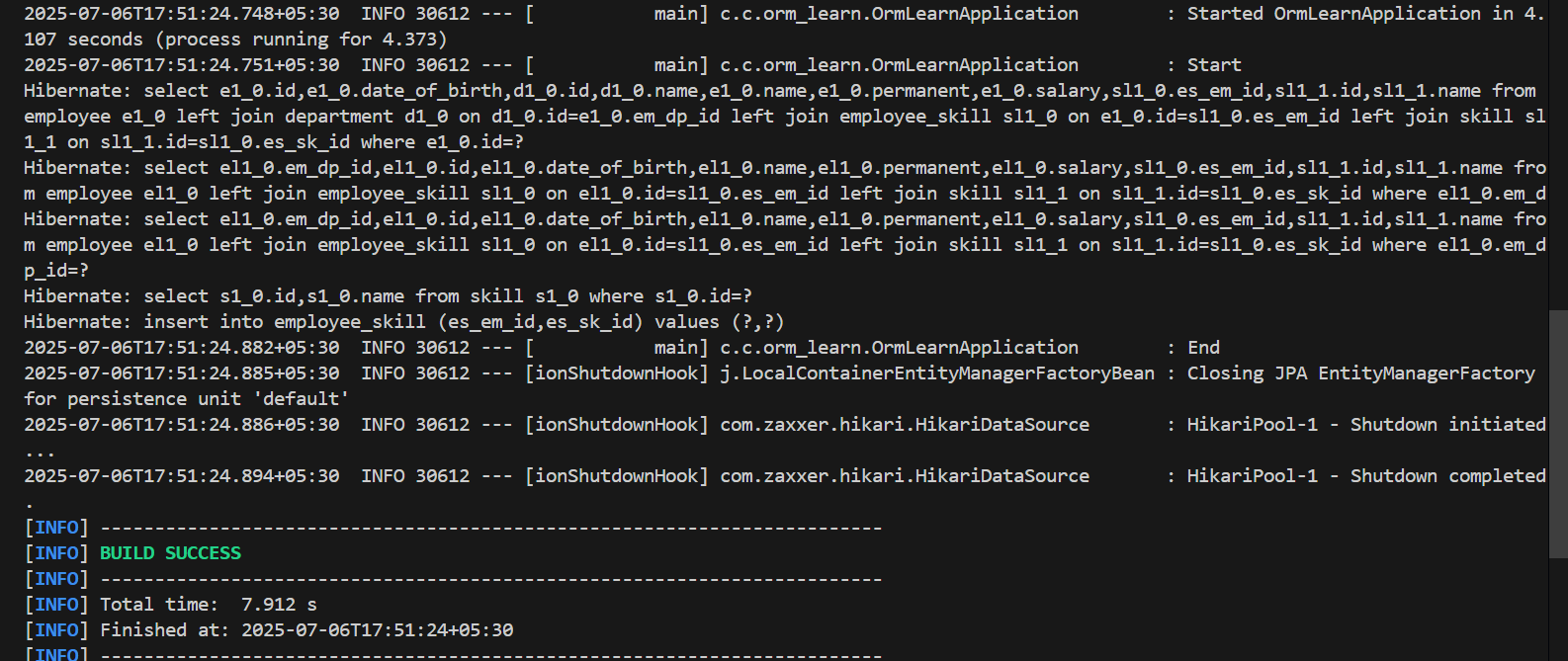
-- Skills

INSERT INTO skill (id, name) VALUES (1, 'Java'), (2, 'Python');

-- Employee Skills (optional initial link)

INSERT INTO employee\_skill (es\_em\_id, es\_sk\_id) VALUES (1, 1);

**Output:**



**Hands on 1**

**Introduction to HQL and JPQL**

**🔹 What is HQL (Hibernate Query Language)?**

* A **database-independent**, **object-oriented** query language provided by **Hibernate**.
* Supports all SQL-like operations: SELECT, UPDATE, DELETE, and **also INSERT** (unlike JPQL).
* Allows querying using **entity names** and their **properties**, not table/column names.

**🔹 What is JPQL (Java Persistence Query Language)?**

* Defined by **JPA** specification.
* Also **object-focused** like HQL but **does not support INSERT**.
* All JPQL queries are valid HQL queries.

Example:

// HQL / JPQL SELECT

String hql = "SELECT e FROM Employee e WHERE e.salary > :salary";

// JPQL UPDATE

String jpql = "UPDATE Employee e SET e.name = :name WHERE e.id = :id";

// JPQL DELETE

String jpqlDelete = "DELETE FROM Employee e WHERE e.id = :id";

// HQL INSERT (not allowed in JPQL)

String hqlInsert = "INSERT INTO Employee (id, name, salary) SELECT id, name, salary FROM TempEmployee";

**Hands on 2**

**Get all permanent employees using HQL** 

**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.Set;

@Entity

public class Employee {

@Id

private int id;

private String name;

private double salary;

private boolean permanent;

@Temporal(TemporalType.DATE)

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "em\_dp\_id")

private Department department;

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(name = "employee\_skill",

joinColumns = @JoinColumn(name = "es\_em\_id"),

inverseJoinColumns = @JoinColumn(name = "es\_sk\_id"))

private Set<Skill> skillList;

}

**Department.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

public class Department {

@Id

private int id;

private String name;

@OneToMany(mappedBy = "department")

private Set<Employee> employeeList;

}

**Skill.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

public class Skill {

@Id

private int id;

private String name;

@ManyToMany(mappedBy = "skillList")

private Set<Employee> employeeList;

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn;

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("SELECT e FROM Employee e LEFT JOIN FETCH e.department d LEFT JOIN FETCH e.skillList WHERE e.permanent = true")

List<Employee> getAllPermanentEmployees();

}

**EmployeeService.java**

package com.cognizant.orm\_learn;

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> getAllPermanentEmployees() {

return employeeRepository.getAllPermanentEmployees();

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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 {

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

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

testGetAllPermanentEmployees();

}

public void testGetAllPermanentEmployees() {

LOGGER.info("Start");

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

LOGGER.debug("Permanent Employees: {}", employees);

employees.forEach(e -> LOGGER.debug("Skills: {}", e.getSkillList()));

LOGGER.info("End");

}

}

**SQL for MySQL**

-- Table creation

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(255)

);

CREATE TABLE skill (

id INT PRIMARY KEY,

name VARCHAR(255)

);

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(255),

salary DOUBLE,

permanent BOOLEAN,

date\_of\_birth DATE,

em\_dp\_id INT,

FOREIGN KEY (em\_dp\_id) REFERENCES department(id)

);

CREATE TABLE employee\_skill (

es\_em\_id INT,

es\_sk\_id INT,

PRIMARY KEY (es\_em\_id, es\_sk\_id),

FOREIGN KEY (es\_em\_id) REFERENCES employee(id),

FOREIGN KEY (es\_sk\_id) REFERENCES skill(id)

);

-- Sample inserts

INSERT INTO department VALUES (1, 'HR'), (2, 'Engineering');

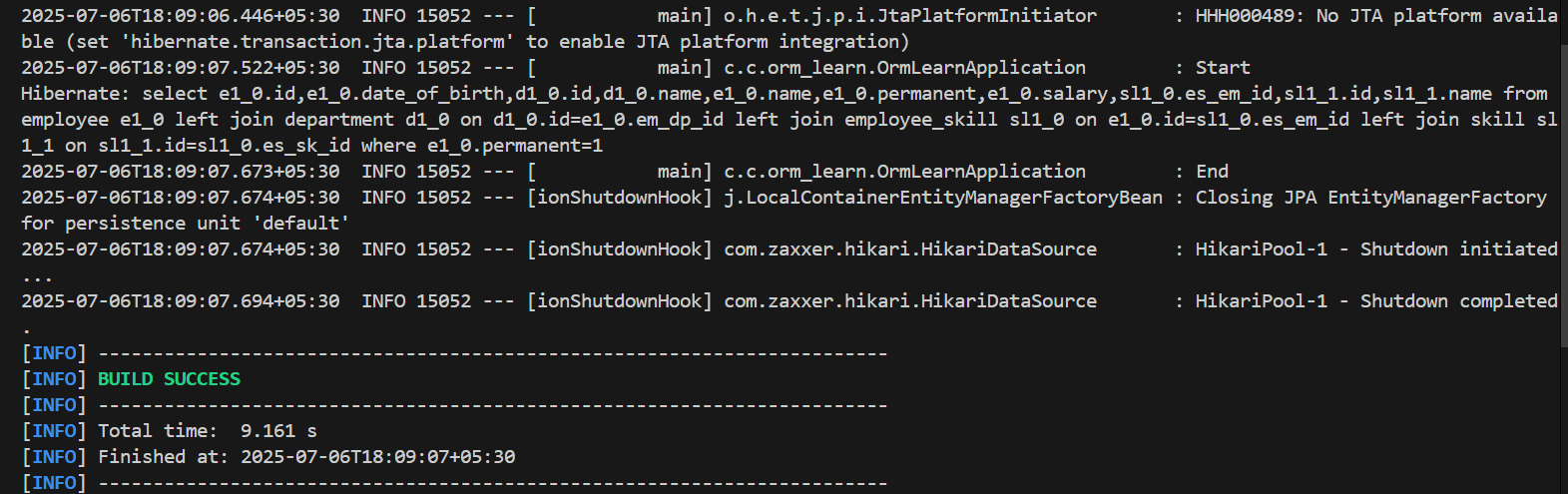
INSERT INTO skill VALUES (1, 'Java'), (2, 'Spring');

INSERT INTO employee VALUES (101, 'Alice', 75000, true, '1990-05-10', 2);

INSERT INTO employee VALUES (102, 'Bob', 55000, false, '1992-08-20', 1);

INSERT INTO employee\_skill VALUES (101, 1), (101, 2);

**Output:**



**Hands on 3**

**Fetch quiz attempt details using HQL**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import java.util.List;

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 AttemptService attemptService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

testGetAttemptDetail();

}

public void testGetAttemptDetail() {

Attempt attempt = attemptService.getAttempt(1);

System.out.println("User: " + attempt.getUser().getName());

int score = 0;

for (AttemptQuestion aq : attempt.getAttemptQuestionList()) {

System.out.println("Question: " + aq.getQuestion().getText());

for (AttemptOption ao : aq.getAttemptOptionList()) {

System.out.println("-- Selected Option: " + ao.getOption().getText());

if (ao.getOption().isCorrect()) {

score++;

}

}

}

System.out.println("Score: " + score);

}

}

**AttemptService.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class AttemptService {

@Autowired

private AttemptRepository attemptRepository {

@Transactional(readOnly = true)

public Attempt getAttempt(int id) {

return attemptRepository.findById(id).orElse(null);

}

}

**AttemptRepository.java**

package com.cognizant.orm\_learn;

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

public interface AttemptRepository extends JpaRepository<Attempt, Integer> {

}

**User.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

public class User {

@Id

private int id;

private String name;

}

**Attempt.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import lombok.Data;

import java.util.List;

@Data

@Entity

public class Attempt {

@Id

private int id;

@ManyToOne

private User user;

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

private List<AttemptQuestion> attemptQuestionList;

}

**AttemptQuestion.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import lombok.Data;

import java.util.List;

@Data

@Entity

public class AttemptQuestion {

@Id

private int id;

@ManyToOne

private Attempt attempt;

@ManyToOne

private Question question;

@OneToMany(mappedBy = "attemptQuestion", fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private List<AttemptOption> attemptOptionList;

}

**AttemptOption.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

public class AttemptOption {

@Id

private int id;

@ManyToOne

private AttemptQuestion attemptQuestion;

@ManyToOne

private Option option;

}

**Question.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import lombok.Data;

import java.util.List;

@Data

@Entity

public class Question {

@Id

private int id;

private String text;

@OneToMany(mappedBy = "question", fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private List<Option> optionList;

}

**Option.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

public class Option {

@Id

private int id;

private String text;

private boolean isCorrect;

@ManyToOne

private Question question;

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**SQL Table Scripts**

CREATE TABLE user (

id INT PRIMARY KEY,

name VARCHAR(100)

);

CREATE TABLE attempt (

id INT PRIMARY KEY,

user\_id INT,

FOREIGN KEY (user\_id) REFERENCES user(id)

);

CREATE TABLE question (

id INT PRIMARY KEY,

text VARCHAR(255)

);

CREATE TABLE question\_option (

id INT PRIMARY KEY,

text VARCHAR(255),

is\_correct BOOLEAN,

question\_id INT,

FOREIGN KEY (question\_id) REFERENCES question(id)

);

CREATE TABLE attempt\_question (

id INT PRIMARY KEY,

attempt\_id INT,

question\_id INT,

FOREIGN KEY (attempt\_id) REFERENCES attempt(id),

FOREIGN KEY (question\_id) REFERENCES question(id)

);

CREATE TABLE attempt\_option (

id INT PRIMARY KEY,

attempt\_question\_id INT,

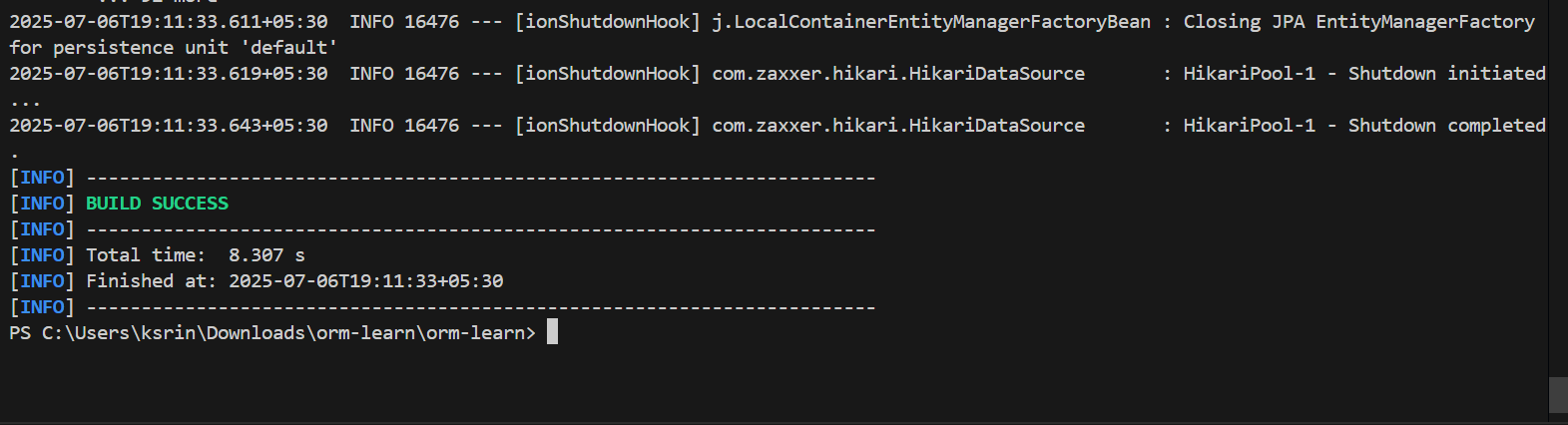
option\_id INT,

FOREIGN KEY (attempt\_question\_id) REFERENCES attempt\_question(id),

FOREIGN KEY (option\_id) REFERENCES question\_option(id)

);

**Output:**



**Hands on 4**

**Get average salary using HQL**

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

spring.datasource.password=your\_mysql\_password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.orm\_learn.service.EmployeeService;

@SpringBootApplication

public class OrmLearnApplication {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

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

OrmLearnApplication app = context.getBean(OrmLearnApplication.class);

app.testGetAverageSalary(1);

}

public void testGetAverageSalary(int deptId) {

double avgSalary = employeeService.getAverageSalary(deptId);

System.out.println("Average salary for department " + deptId + ": " + avgSalary);

}

}

**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

private int id;

private String name;

private double salary;

@ManyToOne

@JoinColumn(name = "dept\_id")

private Department department;

}

**Department.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

@Entity

@Table(name = "department")

public class Department

@Id

private int id;

private String name;

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn;

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

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

import org.springframework.data.repository.CrudRepository;

public interface EmployeeRepository extends CrudRepository<Employee, Integer> {

@Query("SELECT AVG(e.salary) FROM Employee e WHERE e.department.id = :id")

double getAverageSalary(@Param("id") int id);

}

**EmployeeService.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public double getAverageSalary(int deptId) {

return employeeRepository.getAverageSalary(deptId);

}

}

**MySQL**

CREATE DATABASE orm\_learn;

USE orm\_learn;

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(100)

);

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(100),

salary DOUBLE,

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES department(id));

INSERT INTO department VALUES (1, 'HR'), (2, 'IT'), (3, 'Finance');

INSERT INTO employee VALUES

(1, 'Alice', 40000, 1),

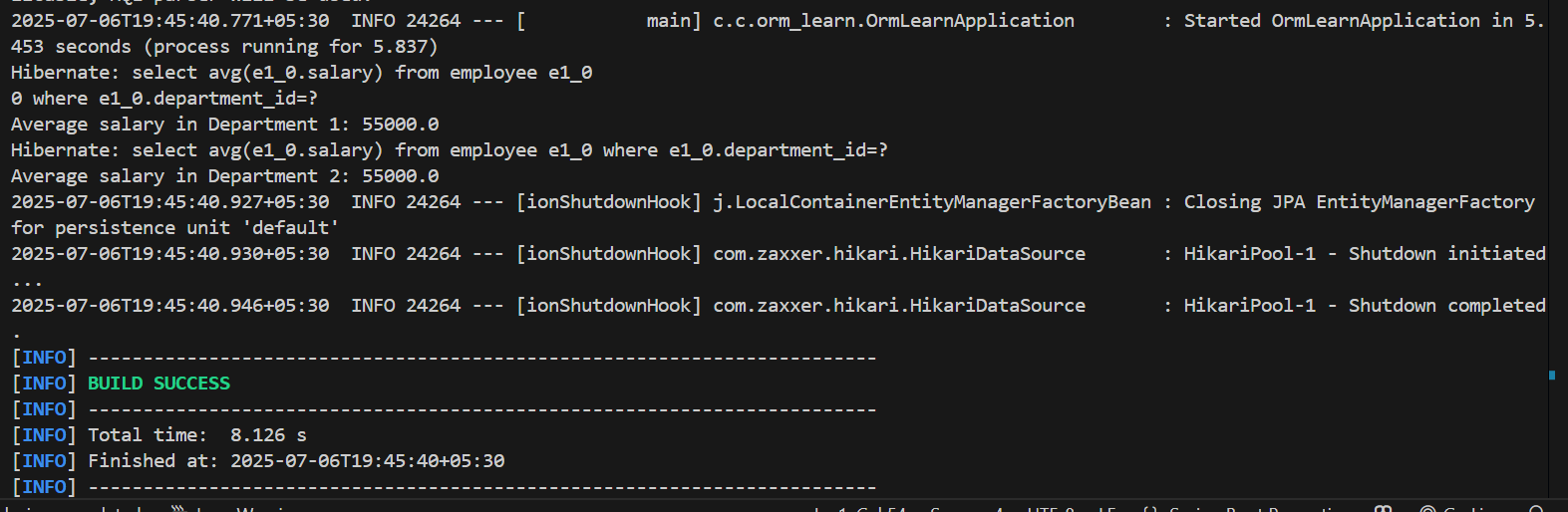
(2, 'Bob', 50000, 1),

(3, 'Carol', 60000, 2),

(4, 'Dave', 55000, 1),

(5, 'Eve', 70000, 3);

**Output:**



**Hands on 5**

**Get all employees using Native Query**

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

spring.datasource.password=your\_mysql\_password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

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

OrmLearnApplication app = context.getBean(OrmLearnApplication.class);

app.testGetAllEmployeesNative();

}

public void testGetAllEmployeesNative() {

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

for (Employee emp : employees) {

System.out.println("ID: " + emp.getId() +

", Name: " + emp.getName() +

", Salary: " + emp.getSalary() +

", Department: " + emp.getDepartment().getName());

}

}

}

**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

private int id;

private String name;

private double salary;

@ManyToOne

@JoinColumn(name = "dept\_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; }

}

**Department.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

@Entity

@Table(name = "department")

public class Department {

@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; }

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn;

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

import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface EmployeeRepository extends CrudRepository<Employee, Integer> {

@Query(value = "SELECT \* FROM employee", nativeQuery = true)

List<Employee> getAllEmployeesNative();

}

**EmployeeService.java**

package com.cognizant.orm\_learn;

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();

}

}

**MySQL**

CREATE DATABASE orm\_learn;

USE orm\_learn;

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(100));

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(100),

salary DOUBLE,

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES department(id)

);

INSERT INTO department VALUES (1, 'HR'), (2, 'IT'), (3, 'Finance');

INSERT INTO employee VALUES

(1, 'Alice', 40000, 1),

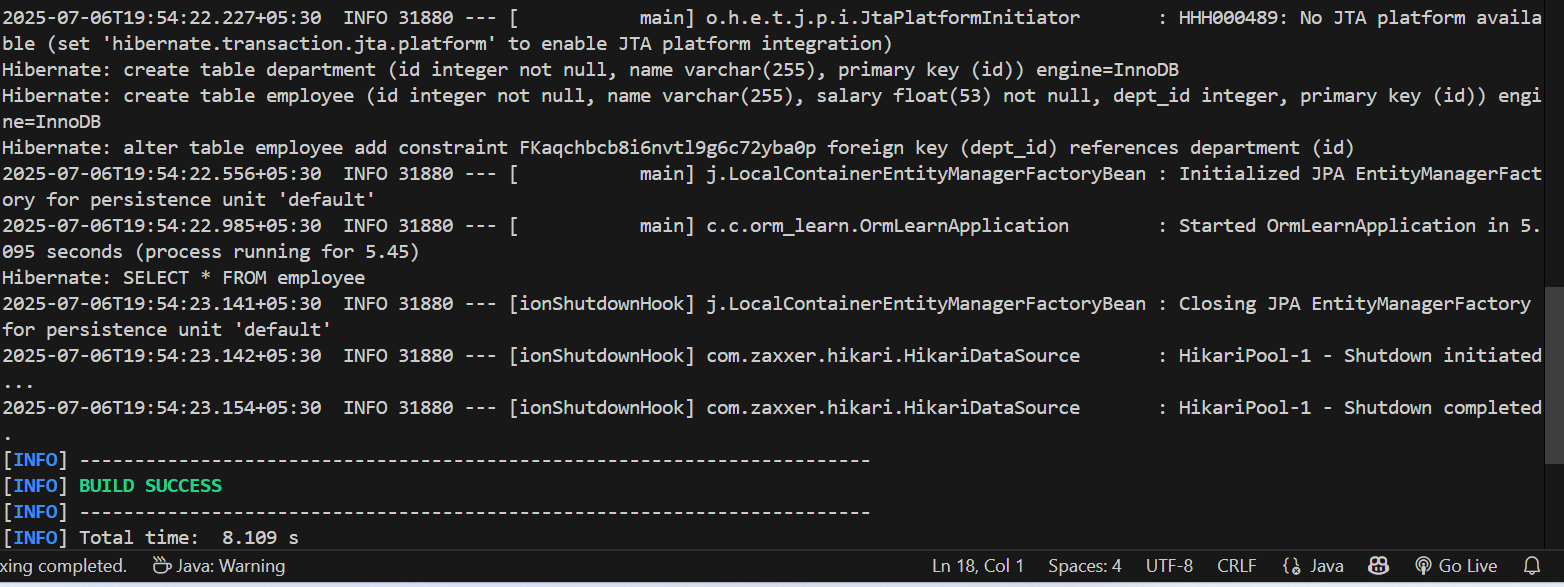
(2, 'Bob', 50000, 1),

(3, 'Carol', 60000, 2),

(4, 'Dave', 55000, 1),

(5, 'Eve', 70000, 3);

**Output:**



**Hands on 6**

**Criteria Query**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

public static void main(String[] args) {

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

ProductService service = context.getBean(ProductService.class);

service.testSearchProducts(); // Criteria Query usage

}

}

**Product.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Product {

@Id

private int id;

private String name;

private double price;

private double cpuSpeed;

private int ramSize;

private String hardDiskSize;

private String operatingSystem;

private double weight;

private double customerReview;

private String cpu;

@Override

public String toString() {

return "Product{" +

"id=" + id +

", name='" + name + '\'' +

", ramSize=" + ramSize +

", OS='" + operatingSystem + '\'' +

", Review=" + customerReview +

'}';

}

}

**ProductRepository.java**

package com.cognizant.orm\_learn;

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

public interface ProductRepository extends JpaRepository<Product, Integer>, ProductRepositoryCustom {

}

**ProductRepositoryCustom.java**

package com.cognizant.orm\_learn;

import java.util.List;

public interface ProductRepositoryCustom {

List<Product> searchProductsWithCriteria(Double minReview, Integer ramSize, String os);

}

**ProductRepositoryImpl.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.EntityManager;

import jakarta.persistence.PersistenceContext;

import jakarta.persistence.criteria.\*;

import org.springframework.stereotype.Repository;

import java.util.ArrayList;

import java.util.List;

@Repository

public class ProductRepositoryImpl implements ProductRepositoryCustom {

@PersistenceContext

private EntityManager em;

@Override

public List<Product> searchProductsWithCriteria(Double minReview, Integer ramSize, String os) {

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Product> cq = cb.createQuery(Product.class);

Root<Product> product = cq.from(Product.class);

List<Predicate> predicates = new ArrayList<>();

if (minReview != null) {

predicates.add(cb.greaterThanOrEqualTo(product.get("customerReview"), minReview));

}

if (ramSize != null) {

predicates.add(cb.equal(product.get("ramSize"), ramSize));

}

if (os != null && !os.isEmpty()) {

predicates.add(cb.equal(product.get("operatingSystem"), os));

}

cq.where(predicates.toArray(new Predicate[0]));

return em.createQuery(cq).getResultList();

} }

**ProductService.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class ProductService {

@Autowired

private ProductRepository productRepository;

public void testSearchProducts() {

List<Product> results = productRepository.searchProductsWithCriteria(4.0, 8, "Windows");

System.out.println("Filtered Products:");

results.forEach(System.out::println);

}

}

**MySQL**

CREATE TABLE product (

id INT PRIMARY KEY,

name VARCHAR(255),

price DOUBLE,

cpu\_speed DOUBLE,

ram\_size INT,

hard\_disk\_size VARCHAR(100),

operating\_system VARCHAR(100),

weight DOUBLE,

customer\_review DOUBLE,

cpu VARCHAR(100)

);

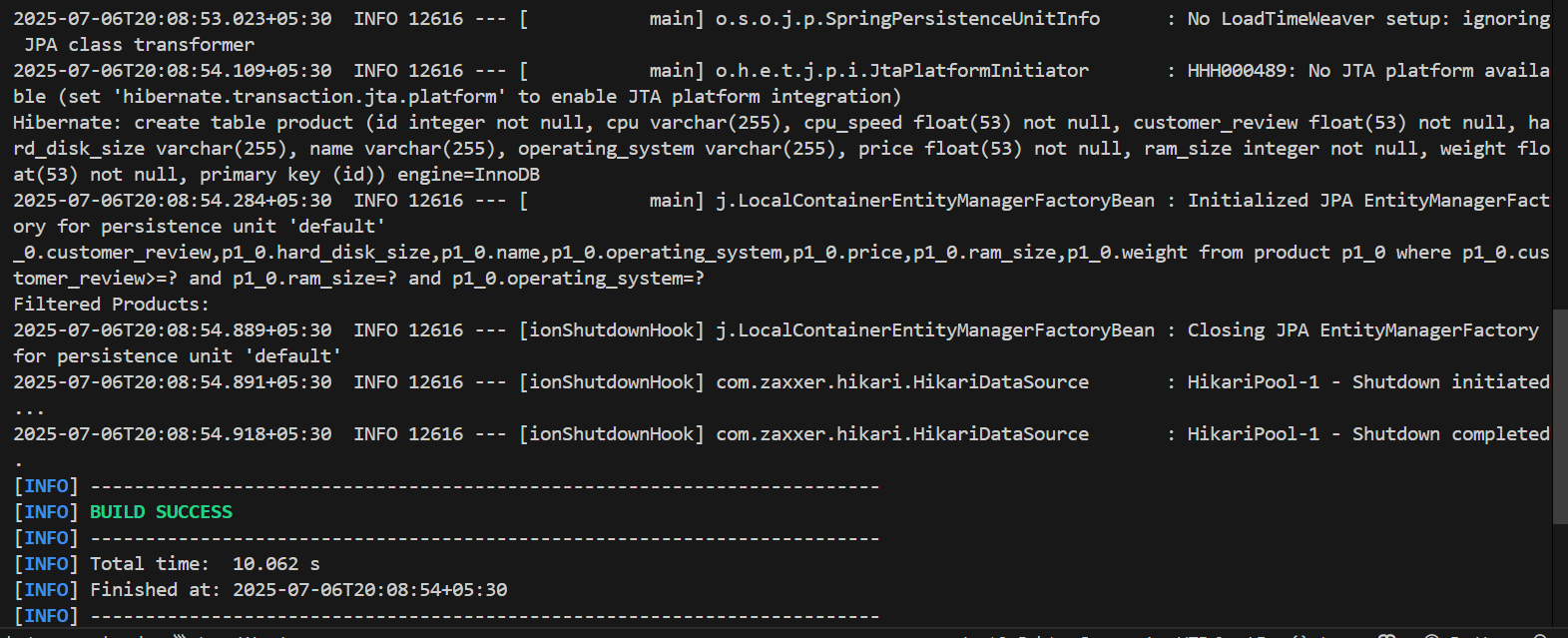
INSERT INTO product VALUES

(1, 'Dell Inspiron', 55000, 2.5, 8, '1TB', 'Windows', 2.2, 4.3, 'Intel i5'),

(2, 'HP Pavilion', 62000, 3.0, 16, '512GB SSD', 'Windows', 1.8, 4.6, 'Intel i7'),

(3, 'MacBook Air', 90000, 3.2, 8, '256GB SSD', 'macOS', 1.2, 4.8, 'Apple M1');

**Output:**



**Exercise 1: Employee Management System - Overview and Setup**

**Employee.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String department;

private double salary;

public Employee(String name, String department, double salary) {

this.name = name;

this.department = department;

this.salary = salary;

}

}

**EmployeeRepository.java**

package com.yashaswini.demo;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**EmployeeService.java**

package com.yashaswini.demo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repository;

public List<Employee> getAllEmployees() {

return repository.findAll();

}

public Employee addEmployee(Employee employee) {

return repository.save(employee);

}

}

**EmployeeController.java**

package com.yashaswini.demo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService service;

@GetMapping

public List<Employee> getAllEmployees() {

return service.getAllEmployees();

}

@PostMapping

public Employee addEmployee(@RequestBody Employee employee) {

return service.addEmployee(employee);

}

}

**EmployeeManagementSystemApplication.java**

package com.yashaswini.demo;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

@Bean

CommandLineRunner run(EmployeeService service) {

return args -> {

service.addEmployee(new Employee("Yashaswini", "IT", 75000));

service.addEmployee(new Employee("Anjali", "HR", 50000));

};

}

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**pom.xml**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

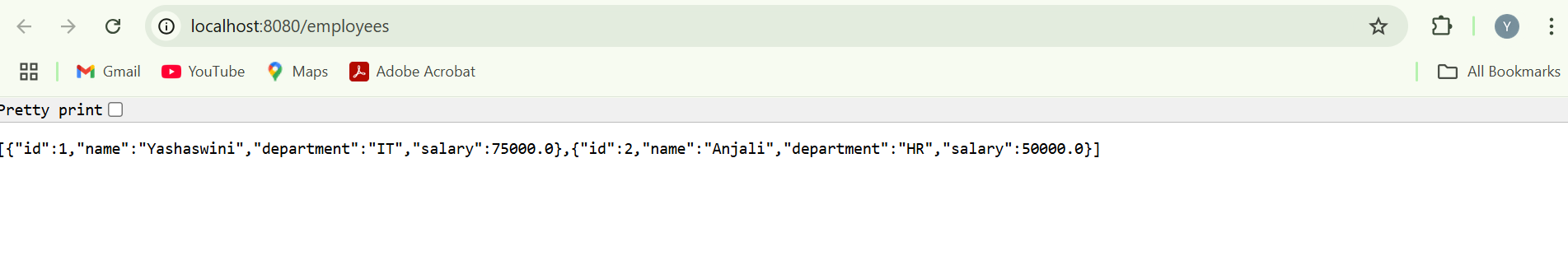
<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

**Output:**



**Exercise 2: Employee Management System - Creating Entities**

**Employee.java**

package com.yashaswini.demo.model;

import jakarta.persistence.\*;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

}

**Department.java**

package com.yashaswini.demo.model;

import jakarta.persistence.\*;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.util.List;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department")

private List<Employee> employees;

}

**EmployeeRepository.java**

package com.yashaswini.demo.repository;

import com.yashaswini.demo.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**DepartmentRepository.java**

package com.yashaswini.demo.repository;

import com.yashaswini.demo.model.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**EmployeeManagementSystemApplication.java**

package com.yashaswini.demo;

import com.yashaswini.demo.model.Department;

import com.yashaswini.demo.model.Employee;

import com.yashaswini.demo.repository.DepartmentRepository;

import com.yashaswini.demo.repository.EmployeeRepository;

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.Arrays;

@SpringBootApplication

public class EmployeeManagementSystemApplication implements CommandLineRunner {

@Autowired

private DepartmentRepository deptRepo;

@Autowired

private EmployeeRepository empRepo;

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

@Override

public void run(String... args) {

Department it = new Department(null, "IT", null);

Department hr = new Department(null, "HR", null);

deptRepo.saveAll(Arrays.asList(it, hr));

Employee emp1 = new Employee(null, "Yashaswini", "yashu@gmail.com", it);

Employee emp2 = new Employee(null, "Anjali", "anjali@gmail.com", hr);

empRepo.saveAll(Arrays.asList(emp1, emp2));

}

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

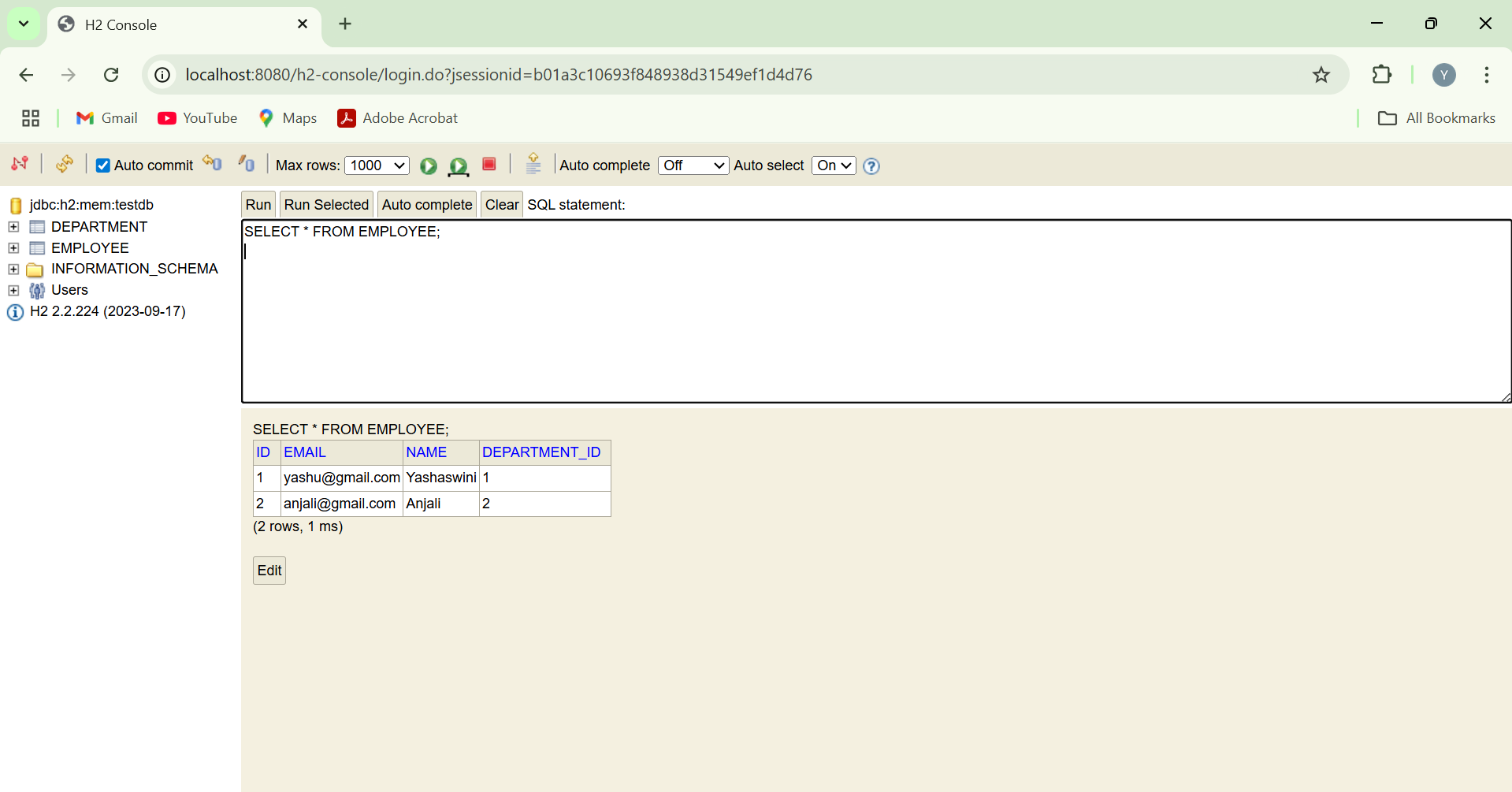
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

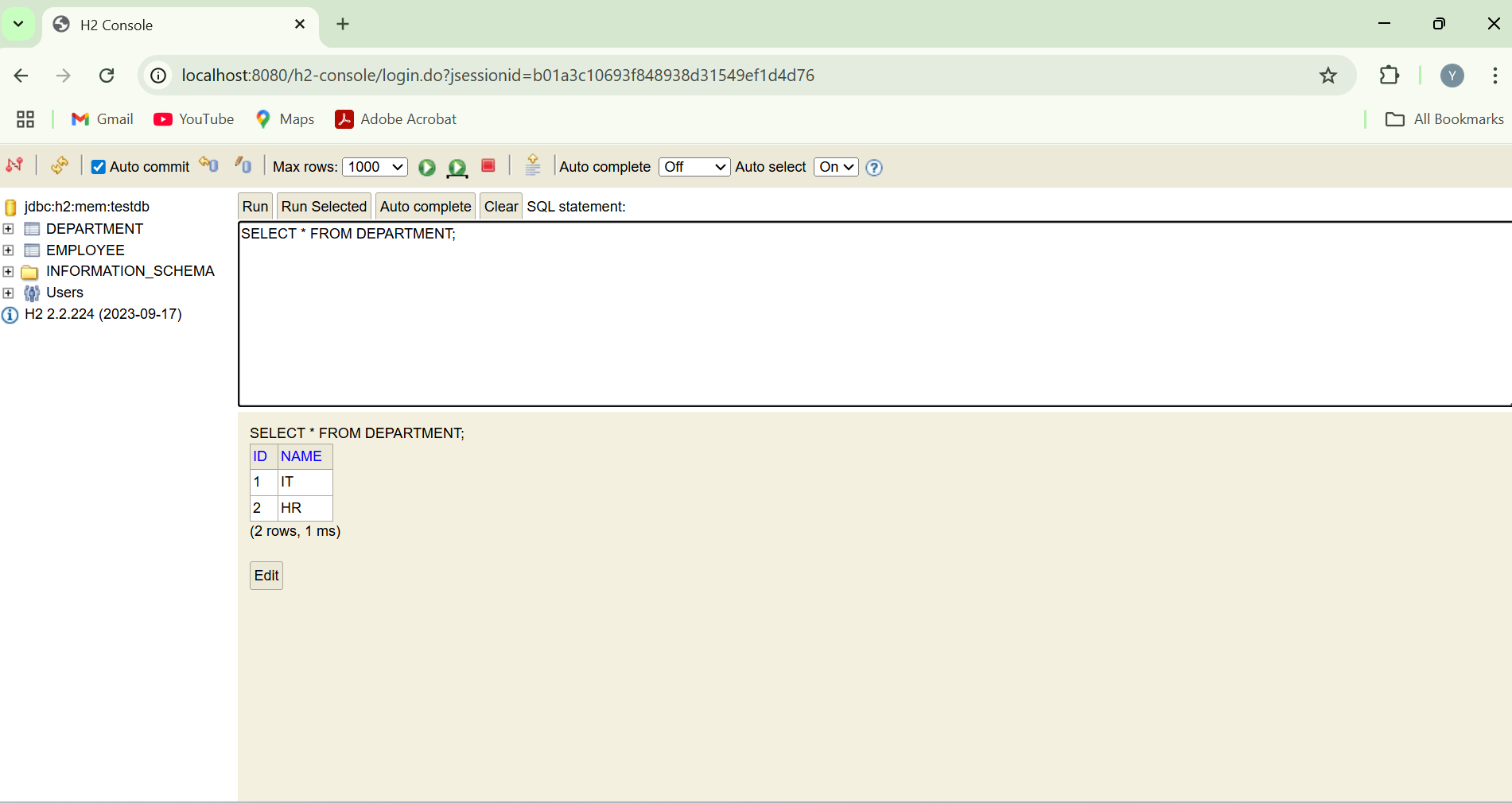
spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.h2.console.enabled=true

**output:**





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

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.yashaswini</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>EmployeeManagementSystem</name>

<packaging>jar</packaging>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=SA

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

**Department.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

private List<Employee> employees;

public Department() {}

public Department(Long id, String name, List<Employee> employees) {

this.id = id;

this.name = name;

this.employees = employees;

}

}

**Employee.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

public Employee() {}

public Employee(Long id, String name, String email, Department department) {

this.id = id;

this.name = name;

this.email = email;

this.department = department;

}

}

**DepartmentRepository.java**

package com.yashaswini.demo;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

Department findByName(String name);

}

**EmployeeRepository.java**

package com.yashaswini.demo;

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<Employee> findByNameContaining(String keyword);

List<Employee> findByDepartmentName(String departmentName); // Uses nested property

}

**EmployeeManagementSystemApplication.java**

package com.yashaswini.demo;

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.Arrays;

import java.util.List;

@SpringBootApplication

public class EmployeeManagementSystemApplication implements CommandLineRunner {

@Autowired

private DepartmentRepository deptRepo;

@Autowired

private EmployeeRepository empRepo;

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

@Override

public void run(String... args) {

Department it = new Department(null, "IT", null);

Department hr = new Department(null, "HR", null);

deptRepo.saveAll(Arrays.asList(it, hr));

Employee emp1 = new Employee(null, "Yashaswini", "yashu@gmail.com", it);

Employee emp2 = new Employee(null, "Anjali", "anjali@gmail.com", hr);

empRepo.saveAll(Arrays.asList(emp1, emp2));

System.out.println("\nEmployees in HR:");

empRepo.findByDepartmentName("HR").forEach(System.out::println);

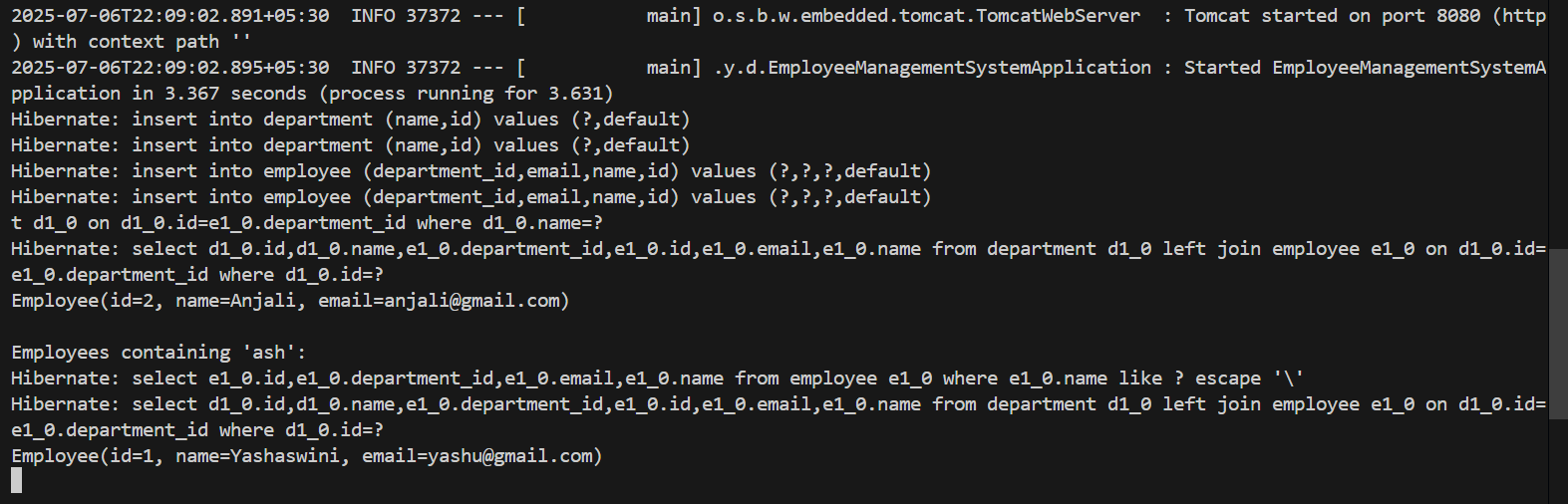
System.out.println("\nEmployees containing 'ash':");

empRepo.findByNameContaining("ash").forEach(System.out::println);

}

}

**Output:**



**Exercise 4: Employee Management System - Implementing CRUD Operations**

**Employee.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "department\_id")

private Department department;

}

**Department.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

import lombok.\*;

import java.util.List;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

private List<Employee> employees;

}

**EmployeeRepository.java**

package com.yashaswini.demo;

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<Employee> findByNameContainingIgnoreCase(String name);

List<Employee> findByDepartmentName(String departmentName);

}

**DepartmentRepository.java**

package com.yashaswini.demo;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

Department findByName(String name);

}

**EmployeeController.java**

package com.yashaswini.demo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

return employeeRepository.findById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@PutMapping("/{id}")

public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee newEmployee) {

return employeeRepository.findById(id)

.map(employee -> {

employee.setName(newEmployee.getName());

employee.setEmail(newEmployee.getEmail());

employee.setDepartment(newEmployee.getDepartment());

return ResponseEntity.ok(employeeRepository.save(employee));

})

.orElse(ResponseEntity.notFound().build());

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {

return employeeRepository.findById(id)

.map(employee -> {

employeeRepository.delete(employee);

return ResponseEntity.noContent().build();

})

.orElse(ResponseEntity.notFound().build());

}

}

**DepartmentController.java**

package com.yashaswini.demo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/departments")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

@GetMapping

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

@GetMapping("/{id}")

public ResponseEntity<Department> getDepartmentById(@PathVariable Long id) {

return departmentRepository.findById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentRepository.save(department);

}

@PutMapping("/{id}")

public ResponseEntity<Department> updateDepartment(@PathVariable Long id, @RequestBody Department newDept) {

return departmentRepository.findById(id)

.map(dept -> {

dept.setName(newDept.getName());

return ResponseEntity.ok(departmentRepository.save(dept));

})

.orElse(ResponseEntity.notFound().build());

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteDepartment(@PathVariable Long id) {

return departmentRepository.findById(id)

.map(dept -> {

departmentRepository.delete(dept);

return ResponseEntity.noContent().build();

})

.orElse(ResponseEntity.notFound().build());

}

}

**EmployeeManagementSystemApplication.java**

package com.yashaswini.demo;

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.Arrays;

@SpringBootApplication

public class EmployeeManagementSystemApplication implements CommandLineRunner {

@Autowired

private DepartmentRepository departmentRepository;

@Autowired

private EmployeeRepository employeeRepository;

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

Department it = new Department(null, "IT", null);

Department hr = new Department(null, "HR", null);

departmentRepository.saveAll(Arrays.asList(it, hr));

Employee emp1 = new Employee(null, "Yashaswini", "yashu@gmail.com", it);

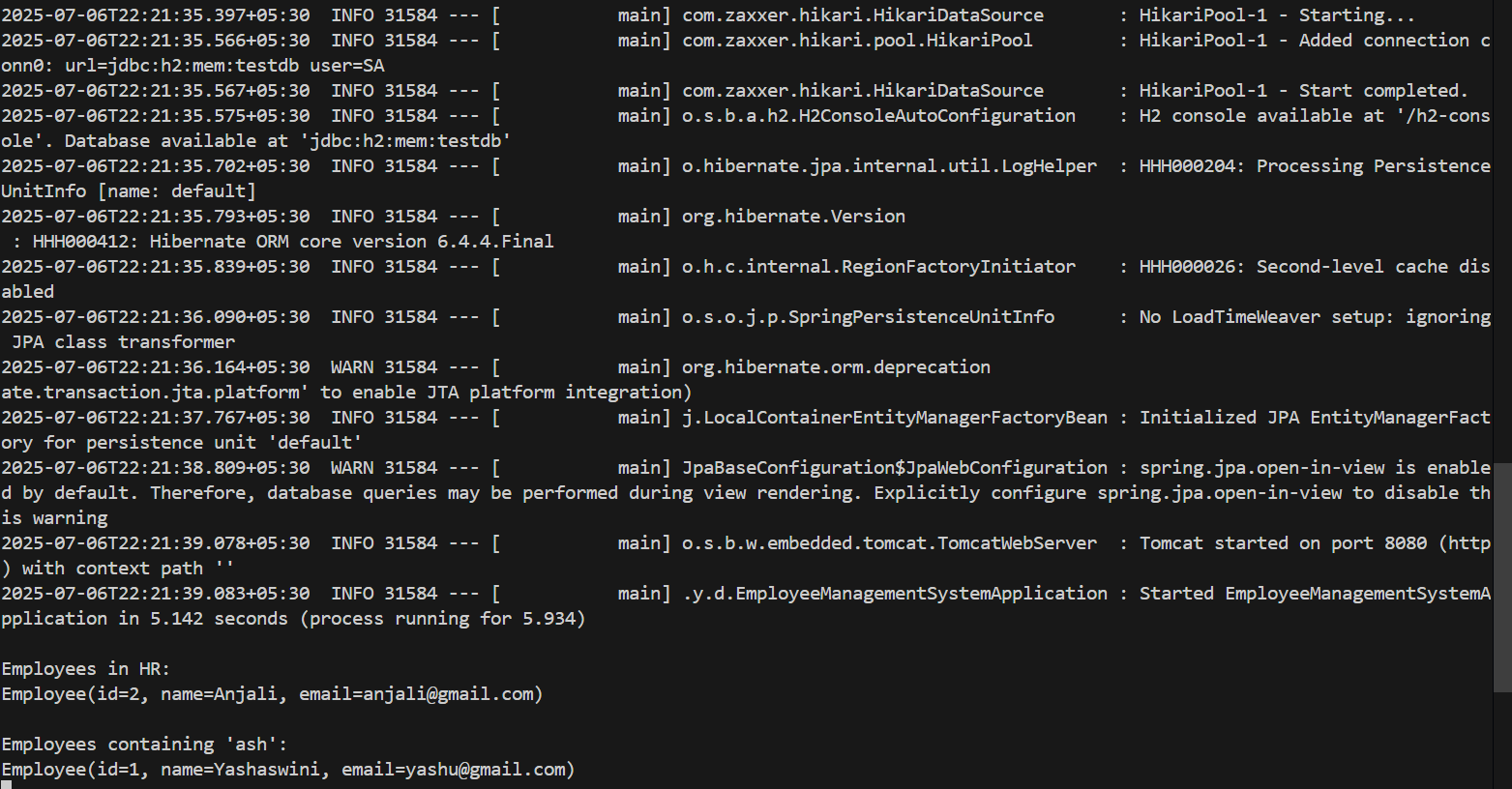
Employee emp2 = new Employee(null, "Anjali", "anjali@gmail.com", hr);

employeeRepository.saveAll(Arrays.asList(emp1, emp2));

}

}

**Output:**



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

**EmployeeManagementSystemApplication.java**

package com.yourcompany.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

}

**Employee.java**

package com.yourcompany.demo;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String email;

@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 String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public Department getDepartment() { return department; }

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

}

**Department.java**

package com.yourcompany.demo;

import jakarta.persistence.\*;

import com.fasterxml.jackson.annotation.JsonIgnore;

import java.util.List;

@Entity

public class Department {

@Id

private int id;

private String name;

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

@JsonIgnore

private List<Employee> employees;

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> getEmployees() { return employees; }

public void setEmployees(List<Employee> employees) { this.employees = employees; }

}

**EmployeeRepository.java**

package com.yourcompany.demo;

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

List<Employee> findByEmailEndingWith(String domain);

}

**DepartmentRepository.java**

package com.yourcompany.demo;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**EmployeeController.java**

package com.yourcompany.demo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private DepartmentRepository departmentRepository;

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@PostMapping("/employees")

public Employee createEmployee(@RequestBody Employee employee) {

Department dept = employee.getDepartment();

if (dept != null && !departmentRepository.existsById(dept.getId())) {

departmentRepository.save(dept);

}

return employeeRepository.save(employee);

}

@GetMapping("/employees/email")

public List<Employee> getEmployeesByEmailDomain(@RequestParam String domain) {

return employeeRepository.findByEmailEndingWith(domain);

}

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**output:**



**Exercise 6: Employee Management System - Implementing Pagination and Sorting**

**EmployeeManagementApplication.java**

package com.yashaswini.demo;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class EmployeeManagementApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementApplication.class, args);

}

@Bean

CommandLineRunner run(EmployeeRepository employeeRepository, DepartmentRepository departmentRepository) {

return args -> {

Department it = new Department();

it.setName("IT");

Department hr = new Department();

hr.setName("HR");

departmentRepository.save(it);

departmentRepository.save(hr);

employeeRepository.save(new Employee("Yashaswini", it, 70000));

employeeRepository.save(new Employee("Rahul", it, 65000));

employeeRepository.save(new Employee("Anjali", hr, 60000));

employeeRepository.save(new Employee("Kiran", hr, 58000));

employeeRepository.save(new Employee("Sneha", it, 62000));

};

}

}

**Employee.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private double salary;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

public Employee() {}

public Employee(String name, Department department, double salary) {

this.name = name;

this.department = department;

this.salary = salary;

}

public Long getId() { return id; }

public String getName() { return name; }

public double getSalary() { return salary; }

public Department getDepartment() { return department; }

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

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

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

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

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + ", department=" + department.getName() + "]";

}

}

**Department.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@JsonIgnore

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

private List<Employee> employees;

public Department() {}

public Department(Long id, String name) {

this.id = id;

this.name = name;

}

public Long getId() { return id; }

public String getName() { return name; }

public List<Employee> getEmployees() { return employees; }

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

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

public void setEmployees(List<Employee> employees) { this.employees = employees; }

}

**EmployeeRepository.java**

package com.yashaswini.demo;

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

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

**DepartmentRepository.java**

package com.yashaswini.demo;

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

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

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/employee\_db

spring.datasource.username=root

spring.datasource.password=yourpassword

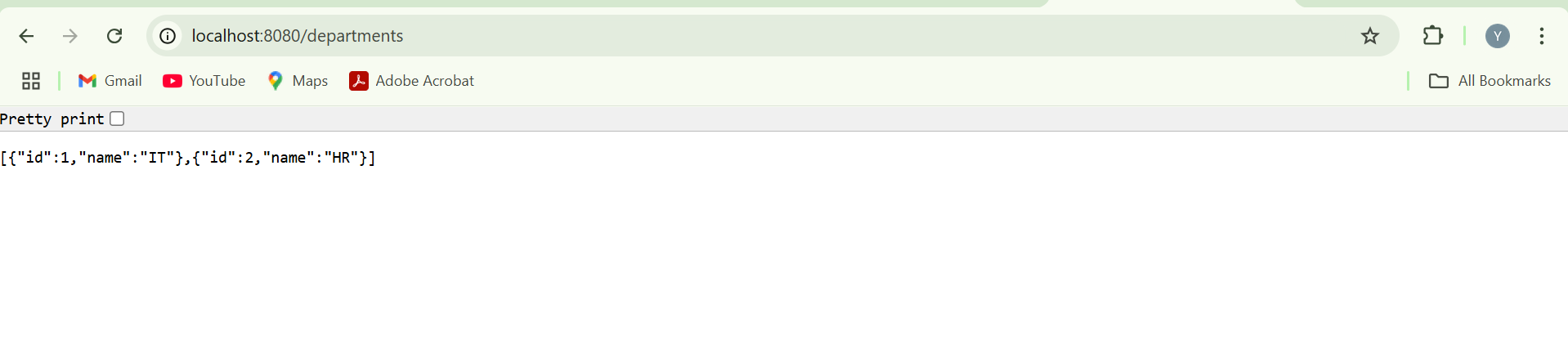
spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**output:**





**Exercise 7: Employee Management System - Enabling Entity Auditing**

**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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.yashaswini</groupId>

<artifactId>employee-management</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

# Enable H2 console

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

# Auditing

spring.jpa.open-in-view=false

**EmployeeManagementApplication.java**

package com.yashaswini.demo;

import jakarta.annotation.PostConstruct;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

import com.yashaswini.demo.model.Department;

import com.yashaswini.demo.model.Employee;

import com.yashaswini.demo.repository.DepartmentRepository;

import com.yashaswini.demo.repository.EmployeeRepository;

import java.time.LocalDateTime;

@SpringBootApplication

@EnableJpaAuditing

public class EmployeeManagementApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementApplication.class, args);

}

@Autowired

private DepartmentRepository departmentRepository;

@Autowired

private EmployeeRepository employeeRepository;

@PostConstruct

public void initData() {

Department dept = new Department();

dept.setName("IT");

departmentRepository.save(dept);

Employee emp = new Employee();

emp.setName("Alice");

emp.setRole("Developer");

emp.setDepartment(dept);

employeeRepository.save(emp);

}

}

**Employee.java**

package com.yashaswini.demo.model;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.Setter;

import org.springframework.data.annotation.CreatedDate;

import org.springframework.data.annotation.LastModifiedDate;

import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.time.LocalDateTime;

@Entity

@Getter

@Setter

@EntityListeners(AuditingEntityListener.class)

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String role;

@ManyToOne

private Department department;

@CreatedDate

private LocalDateTime createdDate;

@LastModifiedDate

private LocalDateTime lastModifiedDate;

}

**Department.java**

package com.yashaswini.demo.model;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.Setter;

import org.springframework.data.annotation.CreatedDate;

import org.springframework.data.annotation.LastModifiedDate;

import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.time.LocalDateTime;

@Entity

@Getter

@Setter

@EntityListeners(AuditingEntityListener.class)

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@CreatedDate

private LocalDateTime createdDate;

@LastModifiedDate

private LocalDateTime lastModifiedDate;

}

**EmployeeRepository.java**

package com.yashaswini.demo.repository;

import com.yashaswini.demo.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

DepartmentRepository.java

package com.yashaswini.demo.repository;

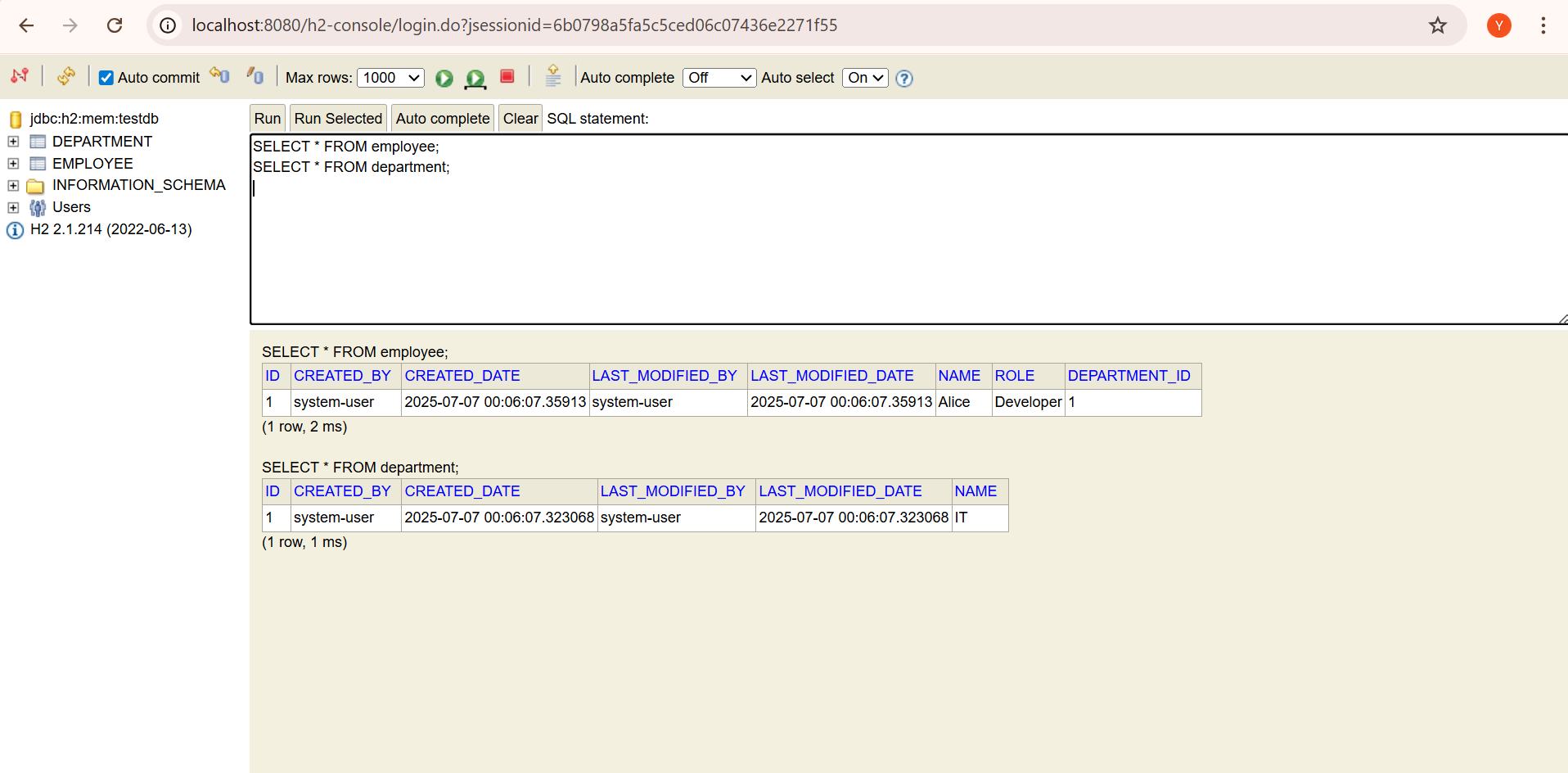
import com.yashaswini.demo.model.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**Output:**



**Exercise 8: Employee Management System - Creating Projections**

**EmployeeManagementApplication.java**

package com.yashaswini.demo;

import jakarta.annotation.PostConstruct;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class EmployeeManagementApplication {

@Autowired

private DepartmentRepository departmentRepository;

@Autowired

private EmployeeRepository employeeRepository;

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementApplication.class, args);

}

@PostConstruct

public void initData() {

Department d1 = new Department("IT");

Department d2 = new Department("HR");

departmentRepository.saveAll(List.of(d1, d2));

employeeRepository.save(new Employee("Alice", "Developer", d1));

employeeRepository.save(new Employee("Bob", "Tester", d1));

employeeRepository.save(new Employee("Carol", "HR Manager", d2));

}

}

**Department.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Getter @Setter @NoArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

public Department(String name) {

this.name = name;

}

}

**Employee.java**

package com.yashaswini.demo;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Getter @Setter @NoArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String role;

@ManyToOne

private Department department;

public Employee(String name, String role, Department department) {

this.name = name;

this.role = role;

this.department = department;

}

}

**EmployeeView.java**

package com.yashaswini.demo;

public interface EmployeeView {

String getName();

String getRole();

}

**EmployeeDTO.java**

package com.yashaswini.demo;

public class EmployeeDTO {

private String name;

private String departmentName;

public EmployeeDTO(String name, String departmentName) {

this.name = name;

this.departmentName = departmentName;

}

public String getName() {

return name;

}

public String getDepartmentName() {

return departmentName;

}

}

**EmployeeRepository.java**

package com.yashaswini.demo;

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

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<EmployeeView> findBy();

@Query("SELECT new com.yashaswini.demo.EmployeeDTO(e.name, e.department.name) FROM Employee e")

List<EmployeeDTO> fetchEmployeeWithDept();

}

**DepartmentRepository.java**

package com.yashaswini.demo;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**EmployeeController.java**

package com.yashaswini.demo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping("/view")

public List<EmployeeView> getEmployeeView() {

return employeeRepository.findBy();

}

@GetMapping("/dto")

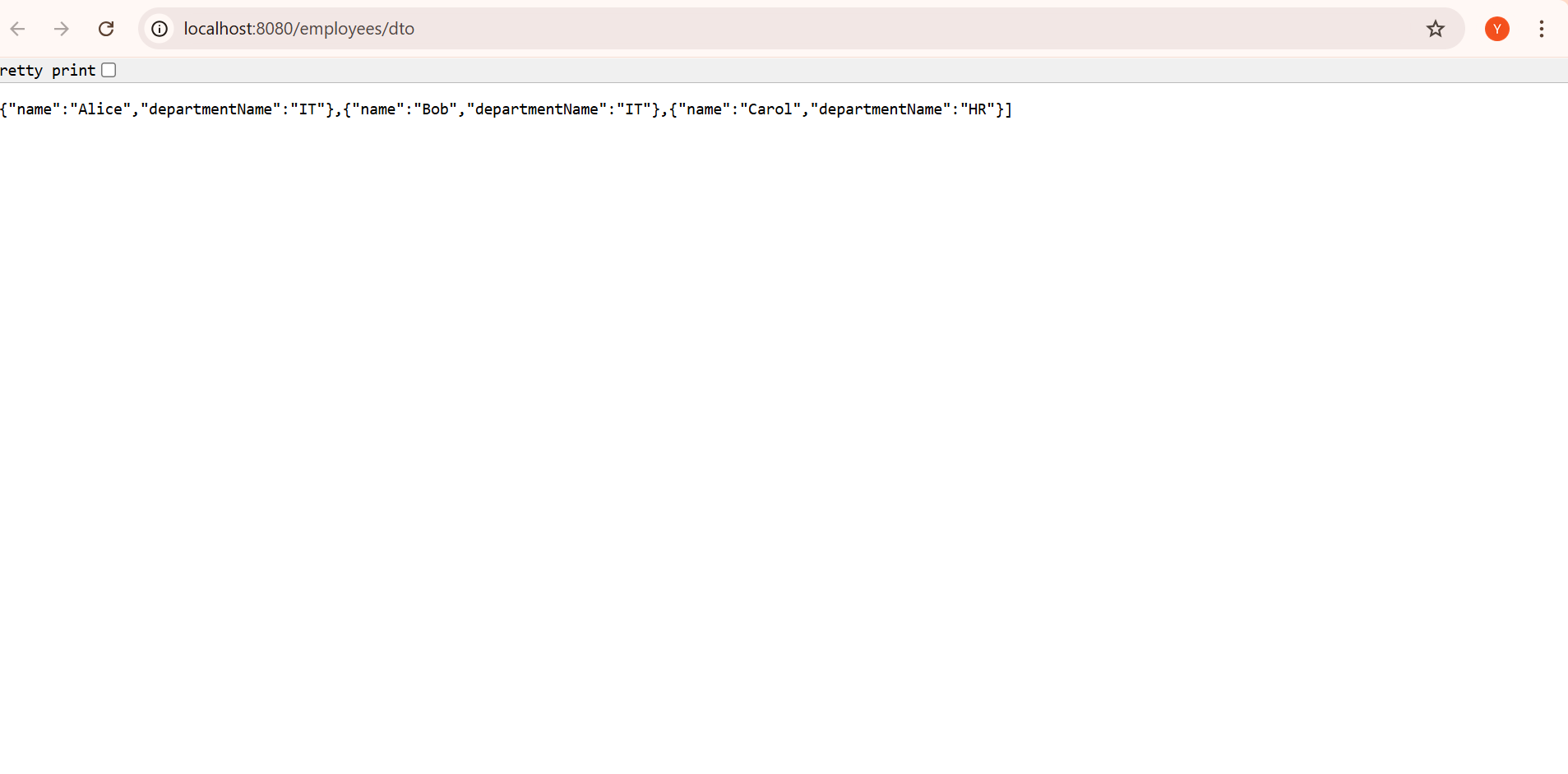
public List<EmployeeDTO> getEmployeeDTO() {

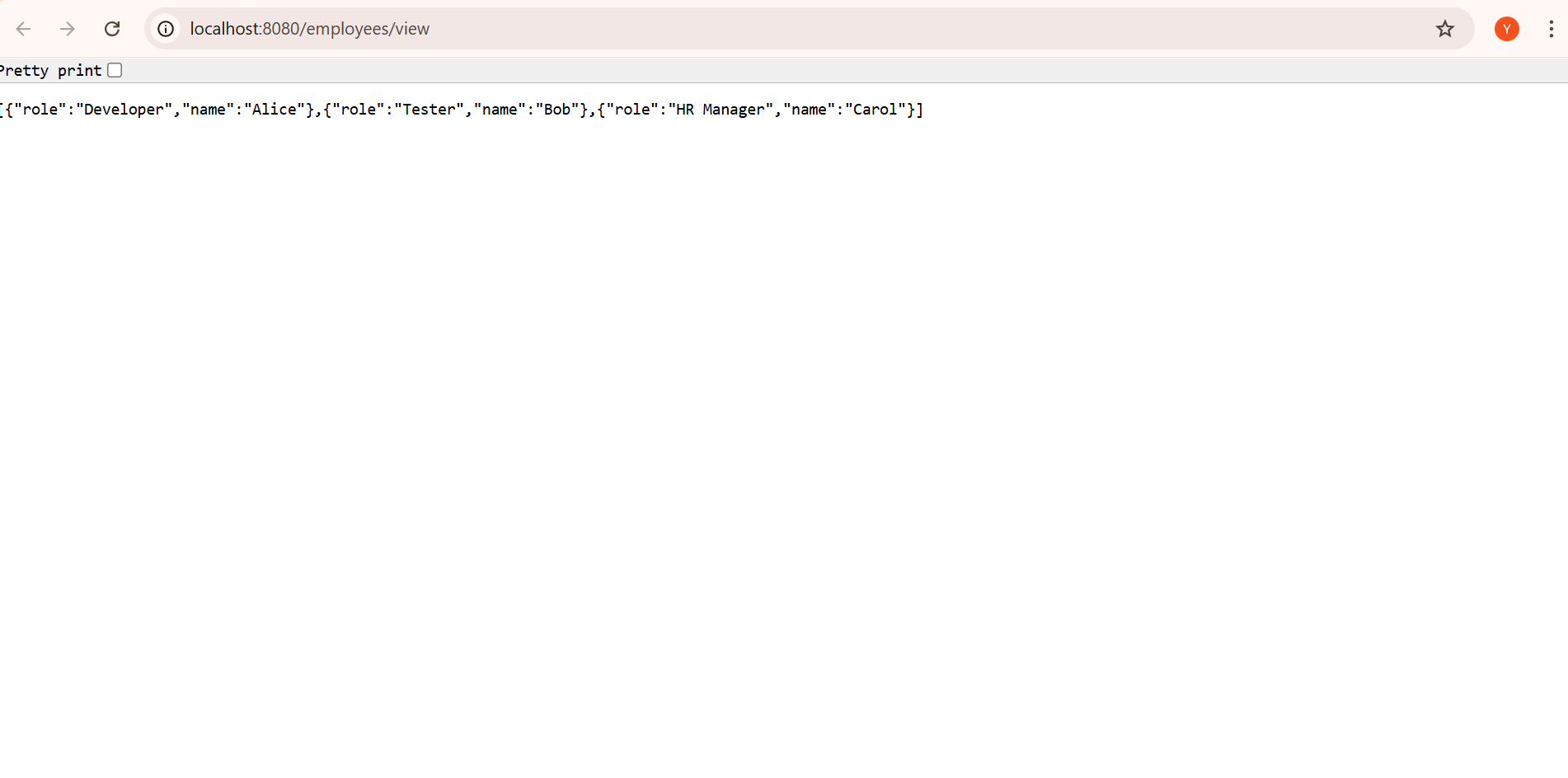
return employeeRepository.fetchEmployeeWithDept();

}

}

**Output:**





**Exercise 9: Employee Management System - Customizing Data Source Configuration**

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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 EmployeeRepository employeeRepo;

@Autowired

private DepartmentRepository departmentRepo;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

System.out.println("Employees:");

employeeRepo.findAll().forEach(emp ->

System.out.println(emp.getId() + " - " + emp.getName()));

System.out.println("Departments:");

departmentRepo.findAll().forEach(dep ->

System.out.println(dep.getId() + " - " + dep.getName()));

}

}

**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "employee")

public class Employee {

@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;

}

}

**Department.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "department")

public class Department {

@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;

}

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**DepartmentRepository.java**

package com.cognizant.orm\_learn;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**SQL**

CREATE DATABASE orm\_learn;

USE orm\_learn;

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(50)

);

INSERT INTO employee (id, name) VALUES

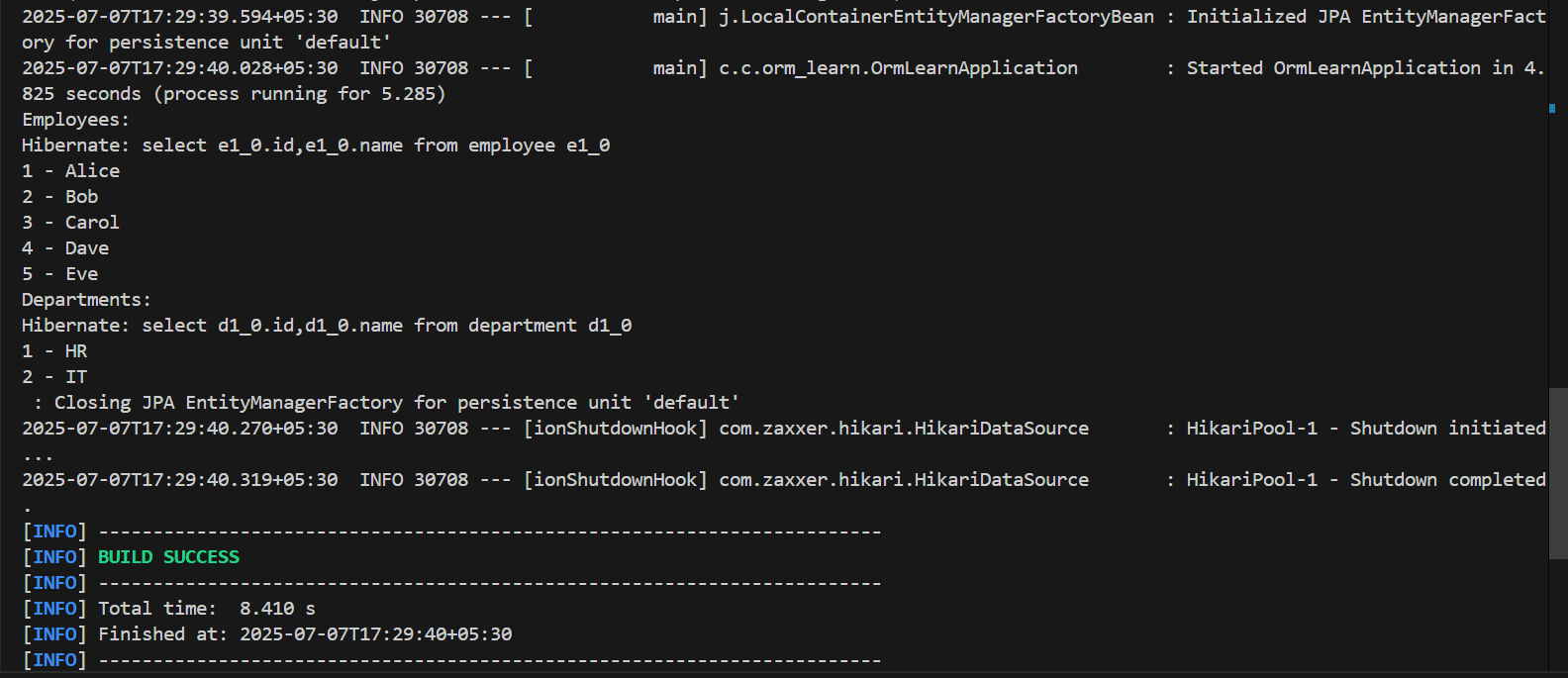
(1, 'Alice'), (2, 'Bob'), (3, 'Carol'),

(4, 'Dave'), (5, 'Eve');

INSERT INTO department (id, name) VALUES

(1, 'HR'), (2, 'IT'), (3, 'Finance');

**Output:**

****

**Exercise 10: Employee Management System - Hibernate-Specific Features**

**Employee.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import org.hibernate.annotations.DynamicInsert;

import org.hibernate.annotations.DynamicUpdate;

@Entity

@DynamicInsert

@DynamicUpdate

@Table(name = "employee")

public class Employee {

@Id

private int id;

@Column(name = "name")

private String name;

public Employee() {}

public Employee(int id, String name) {

this.id = id;

this.name = 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;

}

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import jakarta.persistence.EntityManager;

import jakarta.persistence.PersistenceContext;

import jakarta.transaction.Transactional;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private EmployeeRepository employeeRepository;

@PersistenceContext

private EntityManager entityManager;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

@Transactional

public void run(String... args) throws Exception {

System.out.println("Batch Insertion:");

for (int i = 101; i <= 120; i++) {

Employee emp = new Employee(i, "Emp" + i);

entityManager.persist(emp);

if (i % 10 == 0) {

entityManager.flush();

entityManager.clear();

}

}

System.out.println("All Employees:");

employeeRepository.findAll().forEach(emp ->

System.out.println(emp.getId() + " - " + emp.getName()));

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/department\_db

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.properties.hibernate.generate\_statistics=true

**output:**

