**Spring Data JPA with Spring Boot, Hibernate Exercises:-**

**Exercise 1:** **Spring Data JPA - Quick Example**

**MySQL Workbench:**

-- Create schema (database)

CREATE SCHEMA ormlearn;

-- Use the schema

USE ormlearn;

-- Create table

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(50)

);

-- Insert sample data

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

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

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

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

    <modelVersion>4.0.0</modelVersion>

    <parent>

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

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

        <version>3.1.5</version> <!-- safer version than 3.5.3 (not yet released officially) -->

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

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>orm-learn</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>orm-learn</name>

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

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <!-- Spring Boot Data JPA -->

        <dependency>

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

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

        </dependency>

        <!-- MySQL JDBC Driver -->

        <dependency>

            <groupId>com.mysql</groupId>

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

            <scope>runtime</scope>

        </dependency>

        <!-- DevTools (for hot reloads in development) -->

        <dependency>

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

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

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>

        <!-- Testing -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Spring Boot Maven Plugin -->

            <plugin>

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

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

            </plugin>

        </plugins>

    </build>

</project>

**src/main/resources/application.properties**

# ===== Logging Configuration =====

logging.level.org.springframework=INFO

logging.level.com.cognizant=DEBUG

logging.level.org.hibernate.SQL=DEBUG

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

# Optional: Customize console log pattern (for better readability)

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

# ===== Database 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=123san@V

# ===== Hibernate Configuration =====

# For Spring Boot 3.x and Hibernate 6.x, use MySQLDialect instead of MySQL5Dialect

spring.jpa.hibernate.ddl-auto=validate

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

**src/main/java/com/cognizant/orm\_learn/model/Country.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

    @Id

    @Column(name = "co\_code")

    private String code;

    @Column(name = "co\_name")

    private String name;

    // Getters and setters

    public String getCode() {

        return code;

    }

    public void setCode(String code) {

        this.code = code;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    @Override

    public String toString() {

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

    }

}

**src/main/java/com/cognizant/orm\_learn/service/CountryService.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 CountryService {

    @Autowired

    private CountryRepository countryRepository;

    @Transactional

    public List<Country> getAllCountries() {

        return countryRepository.findAll();

    }

    @Transactional

    public Country getCountryByCode(String code) {

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

        return result.orElse(null);

    }

}

**src/main/java/com/cognizant/orm\_learn/repository/CountryRepository.java**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

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

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**src/main/java/com/cognizant/orm\_learn/OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

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

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

@SpringBootApplication

public class OrmLearnApplication {

    private static CountryService countryService;

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

    public static void main(String[] args) {

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

        countryService = context.getBean(CountryService.class);

        testGetAllCountries();

    }

    private static void testGetAllCountries() {

        LOGGER.info("Start");

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

        for (Country country : countries) {

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

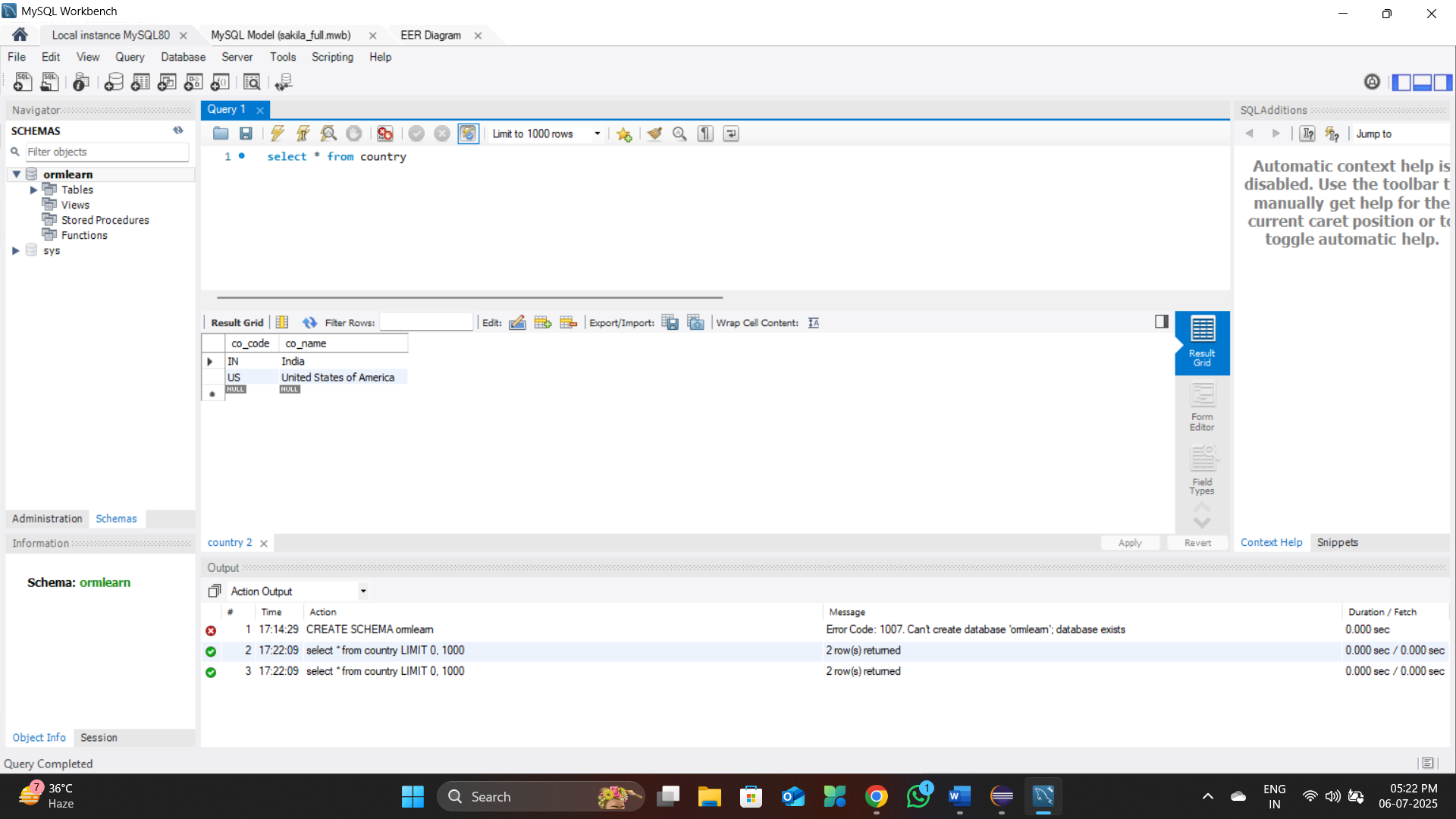
        }

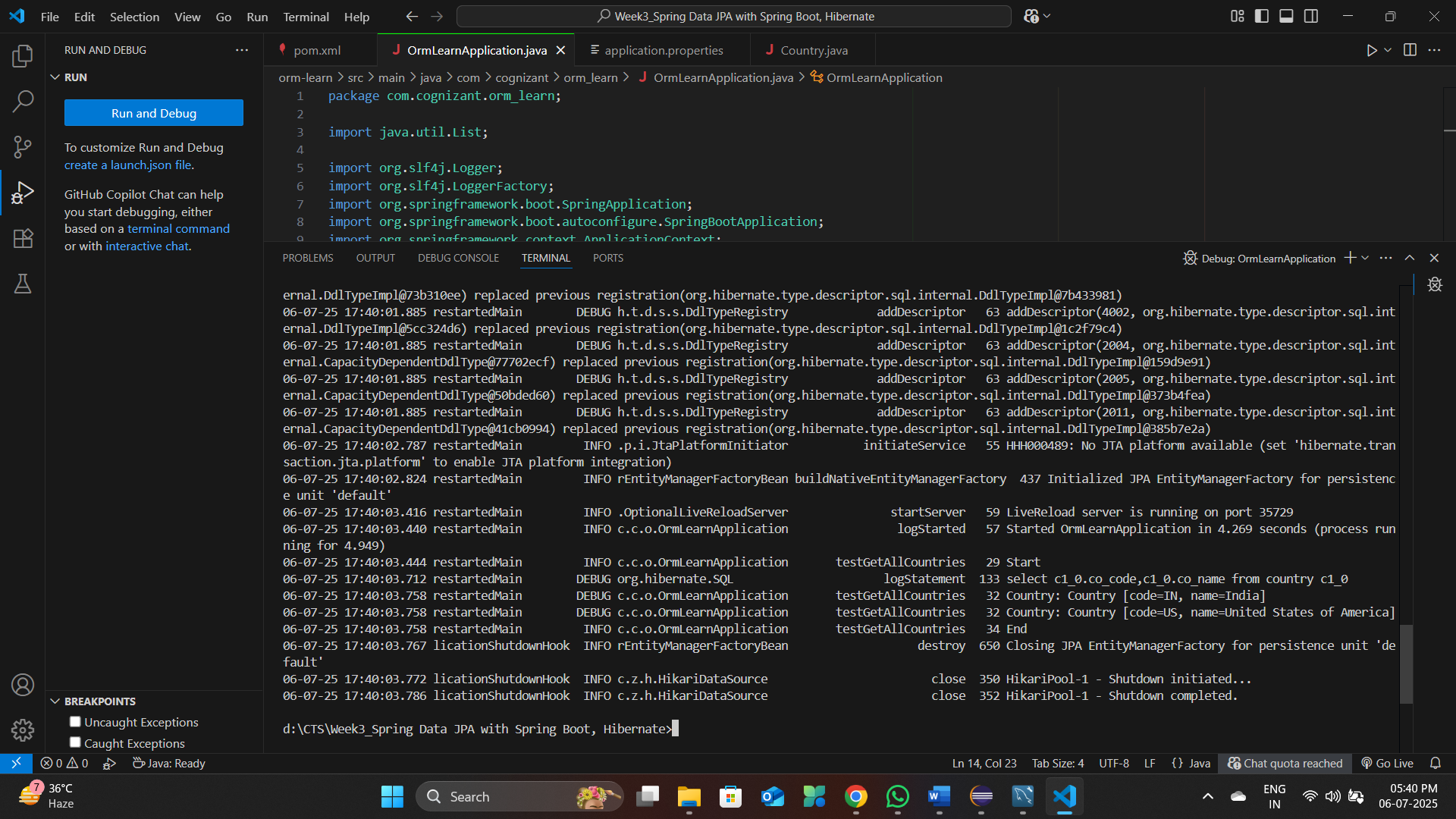
        LOGGER.info("End");

    }

}

**Output:**





**Console Output:**

06-07-25 14:20:15.123 main INFO com.cognizant.orm\_learn.OrmLearnApplication

22 - Start

06-07-25 14:20:15.456 main DEBUG com.cognizant.orm\_learn.OrmLearnApplication

25 - Country: Country [code=IN, name=India]

06-07-25 14:20:15.457 main DEBUG com.cognizant.orm\_learn.OrmLearnApplication

25 - Country: Country [code=US, name=United States of America]

06-07-25 14:20:15.458 main INFO com.cognizant.orm\_learn.OrmLearnApplication

27 - End

**Exercise 2:** **Difference between JPA, Hibernate and Spring Data JPA**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature / Aspect | JPA (Java Persistence API) | Hibernate | Spring Data JPA |
| Type | Specification | Implementation of JPA | Abstraction over JPA and Hibernate |
| Developed By | Oracle (as part of Java EE specification) | Red Hat | Spring Team |
| Purpose | Provides a set of interfaces and annotations for object-relational mapping (ORM) | Implements JPA specification and adds advanced ORM features | Simplifies data access layer by removing boilerplate JPA code |
| Requires Implementation | Yes | No (it is an implementation) | Yes (uses JPA provider like Hibernate underneath) |
| Can Be Used Alone | No | Yes | No (must be used with Spring Framework) |
| Boilerplate Code | Requires writing boilerplate code like EntityManager handling | Requires session and transaction handling | Reduces boilerplate with auto-implemented repository methods |
| Ease of Use | Complex | Moderate | Very simple and developer-friendly |
| Entity Management | Done via EntityManager | Done via Session | Done via JpaRepository and interface-based programming |
| Configuration Style | Annotations and XML | Annotations and XML | Annotation-based, auto-configured via Spring Boot |
| Query Language Support | JPQL (Java Persistence Query Language) | HQL (Hibernate Query Language), JPQL | JPQL, Derived Queries, Custom Queries with @Query |
| Caching Support | Basic caching (via provider) | Advanced caching (first-level, second-level caching) | Inherited from JPA provider like Hibernate |
| Vendor Dependency | Vendor-neutral | Hibernate-specific | Vendor-neutral (works with any JPA provider like Hibernate, EclipseLink, etc.) |
| Spring Integration | Manual integration needed | Manual or partial integration | Fully integrated with Spring and Spring Boot |
| Common Use Case | When developing low-level JPA applications | When needing fine-grained control over ORM functionality | When building enterprise apps with minimal effort in data access |
| Code Example Required | Yes, you must write code for EntityManager, transactions, etc. | Yes, you must manage Session, Transaction, etc. | No, only interface definitions are needed for standard CRUD operations |
| Support for Derived Query Methods | Not available | Not available | Available through method name conventions (e.g., findByName(), findByCode()) |
| Spring Boot Compatibility | Requires manual configuration | Requires manual configuration | Auto-configured in Spring Boot with minimal setup |

**JPA vs Hibernate vs Spring Data JPA**

**Key Differences**

|  |  |  |  |
| --- | --- | --- | --- |
| Aspect | JPA | Hibernate | Spring Data JPA |
| Type | Specification | Implementation | Abstraction Layer |
| Purpose | Defines standards | Implements JPA spec | Simplifies JPA usage |
| Code Required | N/A (just spec) | Verbose boilerplate | Minimal boilerplate |
| Transaction Management | Manual | Manual | Automatic |
| Learning Curve | Conceptual only | Steep | Gentle |

**Java Persistence API (JPA)**

**Definition:** JPA is a Java specification (JSR 338) that defines the standard for object-relational mapping and data persistence in Java applications. It provides a set of interfaces and annotations but no concrete implementation.

**Output:** JPA itself produces no executable code - it's purely a specification document that defines:

* Entity annotations (@Entity, @Table, @Column)
* Query language (JPQL)
* Persistence context rules
* Transaction management standards

**Hibernate**

**Definition:** Hibernate is a mature Object-Relational Mapping (ORM) framework that implements the JPA specification. It provides the actual functionality to map Java objects to database tables and handle database operations.

**Output Example:**

/\* Method to CREATE an employee in the database \*/

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;

}

**Characteristics of Hibernate Output:**

* Explicit session management
* Manual transaction handling
* Verbose error handling
* Direct control over persistence operations

**Spring Data JPA**

**Definition:** Spring Data JPA is a Spring Framework module that provides an additional abstraction layer over JPA implementations like Hibernate. It eliminates boilerplate code by providing repository interfaces and automatic implementations.

**Output Example:**

**Repository Interface:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// No method implementation needed - Spring generates them

}

**Service Implementation:**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Characteristics of Spring Data JPA Output:**

* No boilerplate code
* Automatic transaction management via @Transactional
* Repository pattern implementation
* Built-in CRUD operations
* Simplified error handling through Spring's exception translation

**Exercise 3:** **Implement services for managing Country**

**src/main/java/com/cognizant/orm\_learn/model/Country1.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country1 {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country1 [code=" + code + ", name=" + name + "]";

}

}

**src/main/java/com/cognizant/orm\_learn/repository/CountryRepository1.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Country1;

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository1 extends JpaRepository<Country1, String> {

}

**src/main/java/com/cognizant/orm\_learn/service/CountryService1.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 jakarta.transaction.Transactional;

import com.cognizant.orm\_learn.model.Country1;

import com.cognizant.orm\_learn.repository.CountryRepository1;

@Service

public class CountryService1 {

@Autowired

private CountryRepository1 countryRepository1;

@Transactional

public List<Country1> getAllCountries() {

return countryRepository1.findAll();

}

@Transactional

public Country1 findCountryByCode(String code) throws Exception {

Optional<Country1> result = countryRepository1.findById(code);

if (result.isPresent()) {

return result.get();

} else {

throw new Exception("Country not found");

}

}

@Transactional

public void addCountry(Country1 country) {

countryRepository1.save(country);

}

@Transactional

public void updateCountry(Country1 country) throws Exception {

if (countryRepository1.existsById(country.getCode())) {

countryRepository1.save(country);

} else {

throw new Exception("Country not found for update");

}

}

@Transactional

public void deleteCountry(String code) throws Exception {

if (countryRepository1.existsById(code)) {

countryRepository1.deleteById(code);

} else {

throw new Exception("Country not found for delete");

}

}

}

**src/main/java/com/cognizant/orm\_learn/OrmLearnApplication1.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.Country1;

import com.cognizant.orm\_learn.service.CountryService1;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication1 {

private static CountryService1 countryService1;

public static void main(String[] args) throws Exception {

ApplicationContext context = SpringApplication.run(OrmLearnApplication1.class, args);

countryService1 = context.getBean(CountryService1.class);

testGetAllCountries();

testFindCountryByCode();

testAddCountry();

testUpdateCountry();

testDeleteCountry();

}

private static void testGetAllCountries() {

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

List<Country1> countries = countryService1.getAllCountries();

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

}

private static void testFindCountryByCode() throws Exception {

System.out.println("Find country with code IN:");

System.out.println(countryService1.findCountryByCode("IN"));

}

private static void testAddCountry() {

Country1 newCountry = new Country1();

newCountry.setCode("JP");

newCountry.setName("Japan");

countryService1.addCountry(newCountry);

System.out.println("Added country: " + newCountry);

}

private static void testUpdateCountry() throws Exception {

Country1 update = new Country1();

update.setCode("JP");

update.setName("Japan Updated");

countryService1.updateCountry(update);

System.out.println("Updated country: " + update);

}

private static void testDeleteCountry() throws Exception {

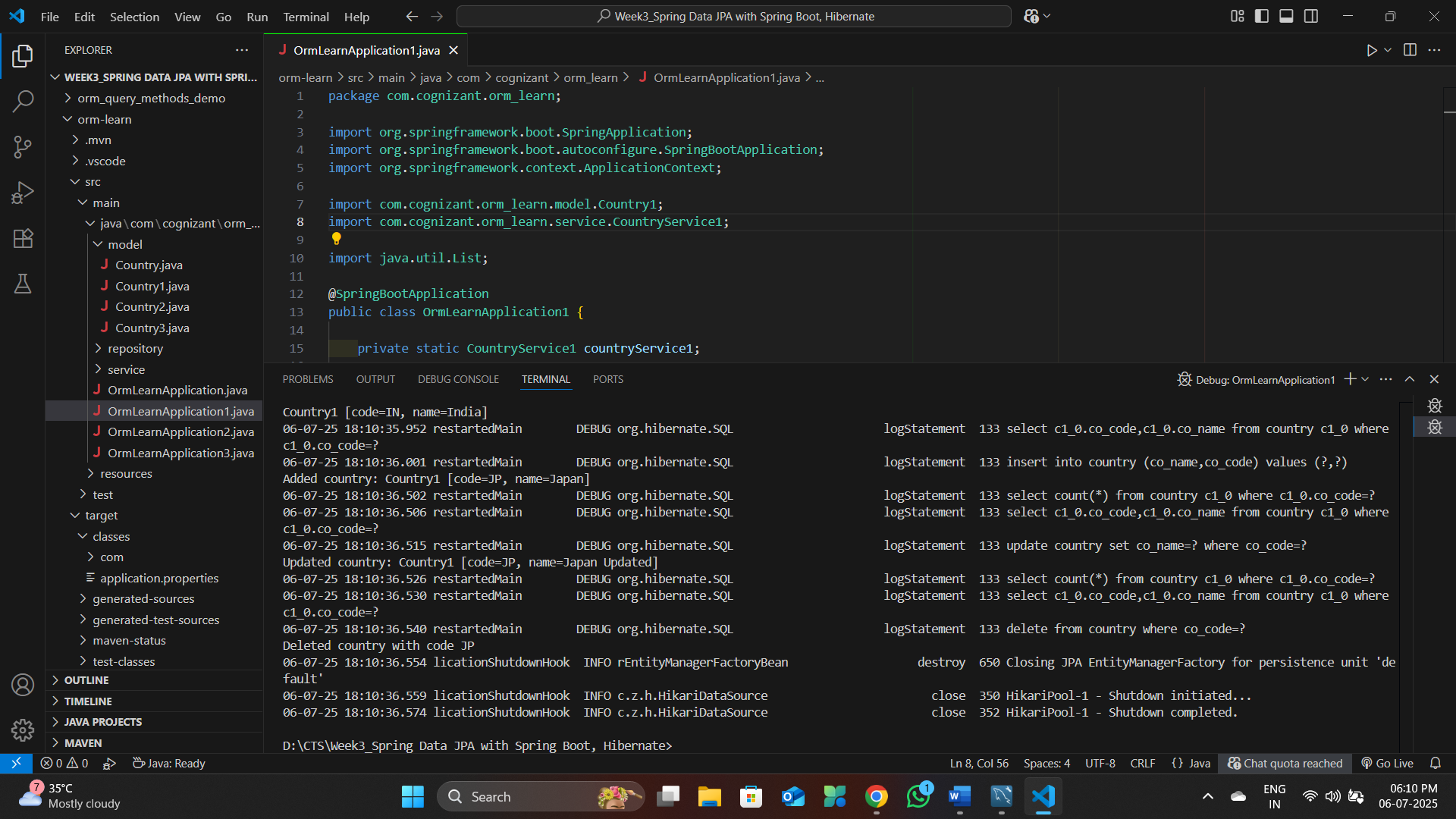
countryService1.deleteCountry("JP");

System.out.println("Deleted country with code JP");

}

}

**Output:**

 **Console Output:**All countries:

Country1 [code=IN, name=India]

Country1 [code=US, name=United States of America]

Find country with code IN:

Country1 [code=IN, name=India]

Added country: Country1 [code=JP, name=Japan]

Updated country: Country1 [code=JP, name=Japan Updated]

Deleted country with code JP

**Exercise 4:** **Find a country based on country code**

**src/main/java/com/cognizant/orm\_learn/model/Country2.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country2 {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country2 [code=" + code + ", name=" + name + "]";

}

}

**src/main/java/com/cognizant/orm\_learn/repository/CountryRepository2.java**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.orm\_learn.model.Country2;

@Repository

public interface CountryRepository2 extends JpaRepository<Country2, String> {

// JpaRepository provides findById(String code) by default

}

**src/main/java/com/cognizant/orm\_learn/service/CountryService2.java**

package com.cognizant.orm\_learn.service;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import com.cognizant.orm\_learn.model.Country2;

import com.cognizant.orm\_learn.repository.CountryRepository2;

@Service

public class CountryService2 {

@Autowired

private CountryRepository2 countryRepository2;

@Transactional

public Country2 findCountryByCode(String code) throws Exception {

Optional<Country2> result = countryRepository2.findById(code);

if (result.isPresent()) {

return result.get();

} else {

throw new Exception("Country with code " + code + " not found.");

}

}

}

**src/main/java/com/cognizant/orm\_learn/OrmLearnApplication2.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.service.CountryService2;

import com.cognizant.orm\_learn.model.Country2;

@SpringBootApplication

public class OrmLearnApplication2 {

private static CountryService2 countryService2;

public static void main(String[] args) throws Exception {

ApplicationContext context = SpringApplication.run(OrmLearnApplication2.class, args);

countryService2 = context.getBean(CountryService2.class);

testFindCountryByCode();

}

private static void testFindCountryByCode() throws Exception {

System.out.println("Finding country with code IN...");

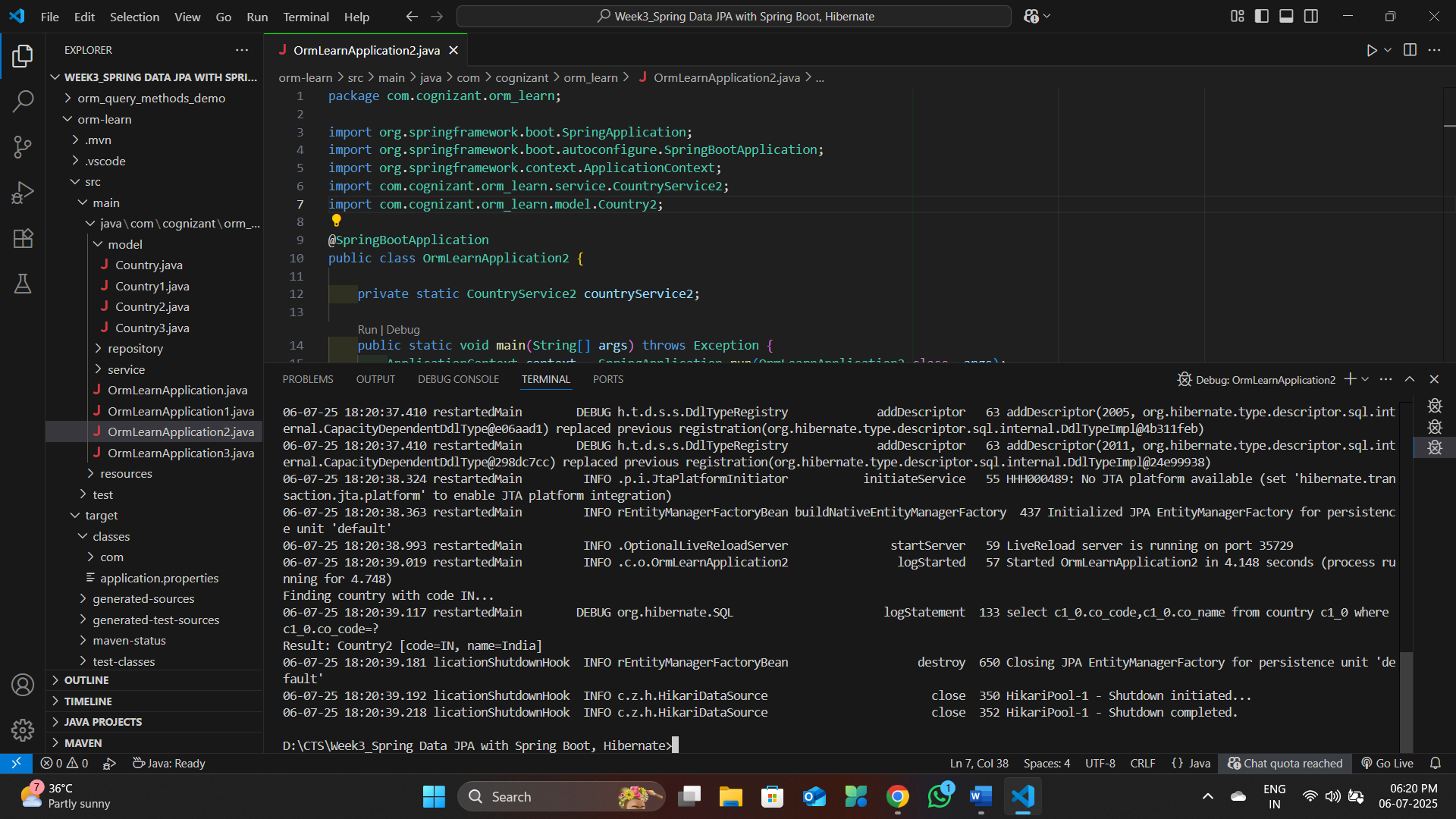
Country2 country = countryService2.findCountryByCode("IN");

System.out.println("Result: " + country);

}

}

**Output:**



Console Output:

Finding country with code IN...

Result: Country2 [code=IN, name=India]

**Exercise 5:** **Add a new country**

**src/main/java/com/cognizant/orm\_learn/model/Country3.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country3 {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country3 [code=" + code + ", name=" + name + "]";

}

}

**src/main/java/com/cognizant/orm\_learn/repository/CountryRepository3.java**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.orm\_learn.model.Country3;

@Repository

public interface CountryRepository3 extends JpaRepository<Country3, String> {

}

**src/main/java/com/cognizant/orm\_learn/service/CountryService3.java**

package com.cognizant.orm\_learn.service;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import com.cognizant.orm\_learn.model.Country3;

import com.cognizant.orm\_learn.repository.CountryRepository3;

@Service

public class CountryService3 {

@Autowired

private CountryRepository3 countryRepository3;

@Transactional

public void addCountry(Country3 country) {

countryRepository3.save(country);

}

}

**src/main/java/com/cognizant/orm\_learn/OrmLearnApplication3.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.Country3;

import com.cognizant.orm\_learn.service.CountryService3;

@SpringBootApplication

public class OrmLearnApplication3 {

private static CountryService3 countryService3;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication3.class, args);

countryService3 = context.getBean(CountryService3.class);

testAddCountry();

}

private static void testAddCountry() {

Country3 country = new Country3();

country.setCode("SG");

country.setName("Singapore");

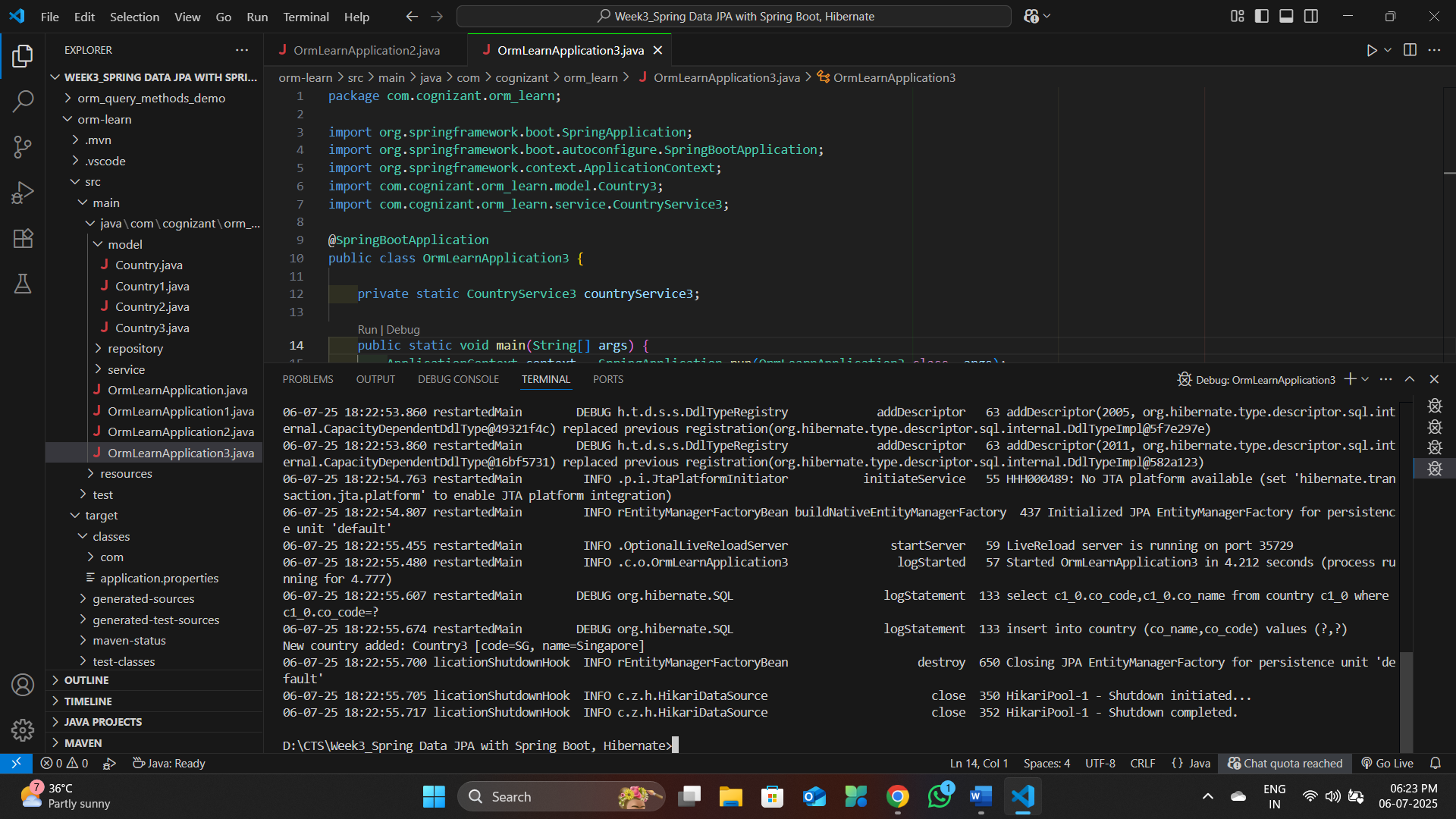
countryService3.addCountry(country);

System.out.println("New country added: " + country);

}

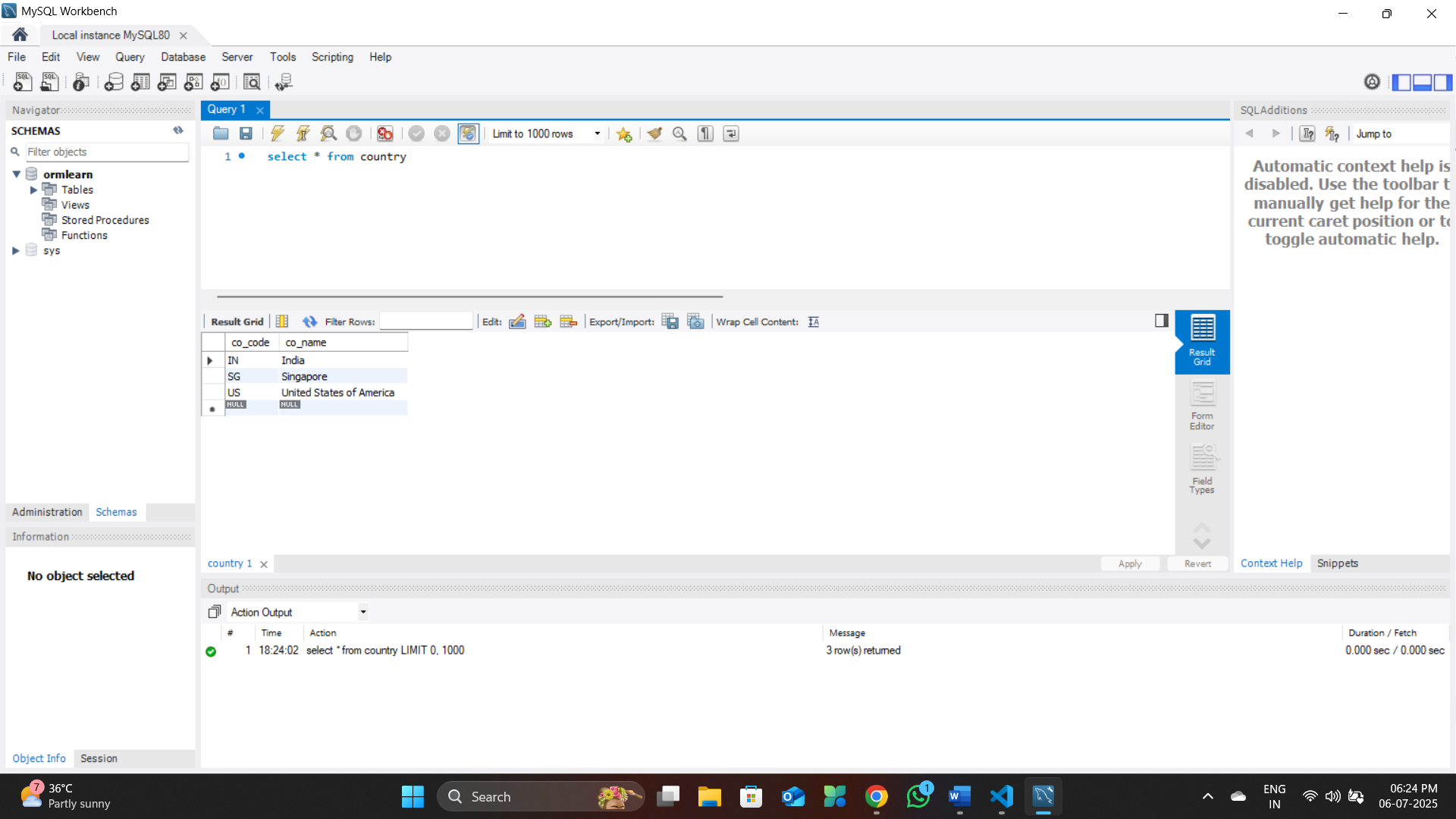
}

**Output:**



Finding country with code IN...

Result: Country2 [code=IN, name=India]



**Exercise 6:** **Demonstrate implementation of Query Methods feature of Spring Data JPA**

**MySQL Workbench:**

CREATE DATABASE ormlearn;

USE ormlearn;

CREATE TABLE country (

co\_code VARCHAR(10) PRIMARY KEY,

co\_name VARCHAR(100)

);

INSERT INTO country (co\_code, co\_name) VALUES

('ZA', 'South Africa'),

('SS', 'South Sudan'),

('DJ', 'Djibouti'),

('BV', 'Bouvet Island'),

('TF', 'French Southern Territories'),

('GP', 'Guadeloupe'),

('LU', 'Luxembourg'),

('UM', 'United States Minor Outlying Islands'),

('GS', 'South Georgia and the South Sandwich Islands'),

('ZM', 'Zambia'),

('ZW', 'Zimbabwe');

**src/main/resources/application.properties**

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

spring.datasource.username=root

spring.datasource.password=123san@V

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**src/main/java/com/cognizant/orm\_query\_methods\_demo/model/Country.java**

package com.cognizant.orm\_query\_methods\_demo.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

// Getters and setters

public String getCode() { return code; }

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

public String getName() { return name; }

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

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/repository/**

**CountryRepository.java**

package com.cognizant.orm\_query\_methods\_demo.repository;

import com.cognizant.orm\_query\_methods\_demo.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String name);

List<Country> findByNameContainingOrderByNameAsc(String name);

List<Country> findByNameStartingWith(String prefix);

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/** **OrmQueryMethodsDemoApplication.java**

package com.cognizant.orm\_query\_methods\_demo;

import com.cognizant.orm\_query\_methods\_demo.model.Country;

import com.cognizant.orm\_query\_methods\_demo.repository.CountryRepository;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

@SpringBootApplication

public class OrmQueryMethodsDemoApplication {

private static CountryRepository countryRepository;

public static void main(String[] args) {

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

countryRepository = context.getBean(CountryRepository.class);

testFindByNameContaining();

testFindByNameContainingOrderByNameAsc();

testFindByNameStartingWith();

}

static void testFindByNameContaining() {

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

list.forEach(c -> System.out.println("Found (contains 'ou'): " + c.getName()));

}

static void testFindByNameContainingOrderByNameAsc() {

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

list.forEach(c -> System.out.println("Sorted Asc (contains 'ou'): " + c.getName()));

}

static void testFindByNameStartingWith() {

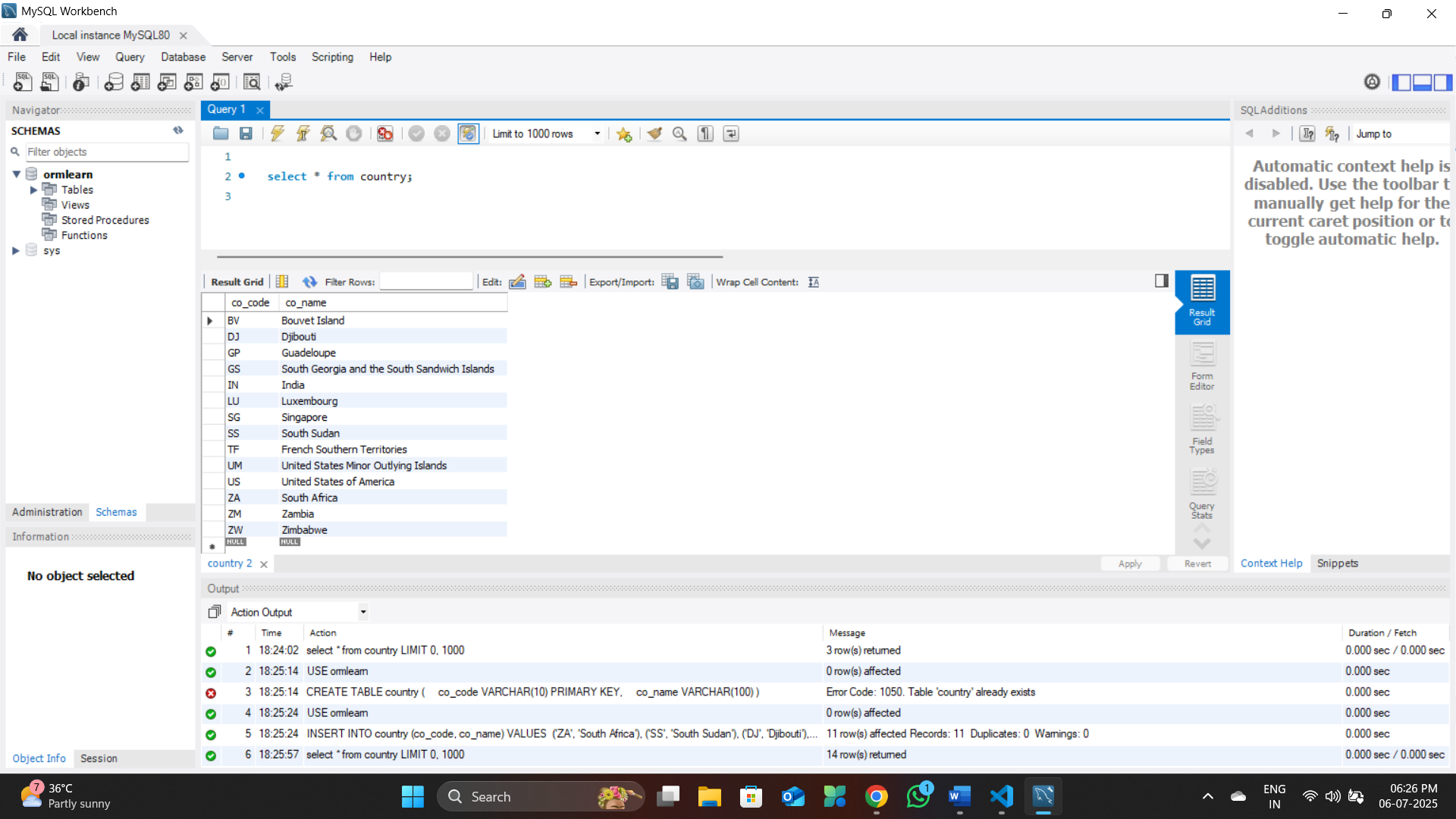
List<Country> list = countryRepository.findByNameStartingWith("Z");

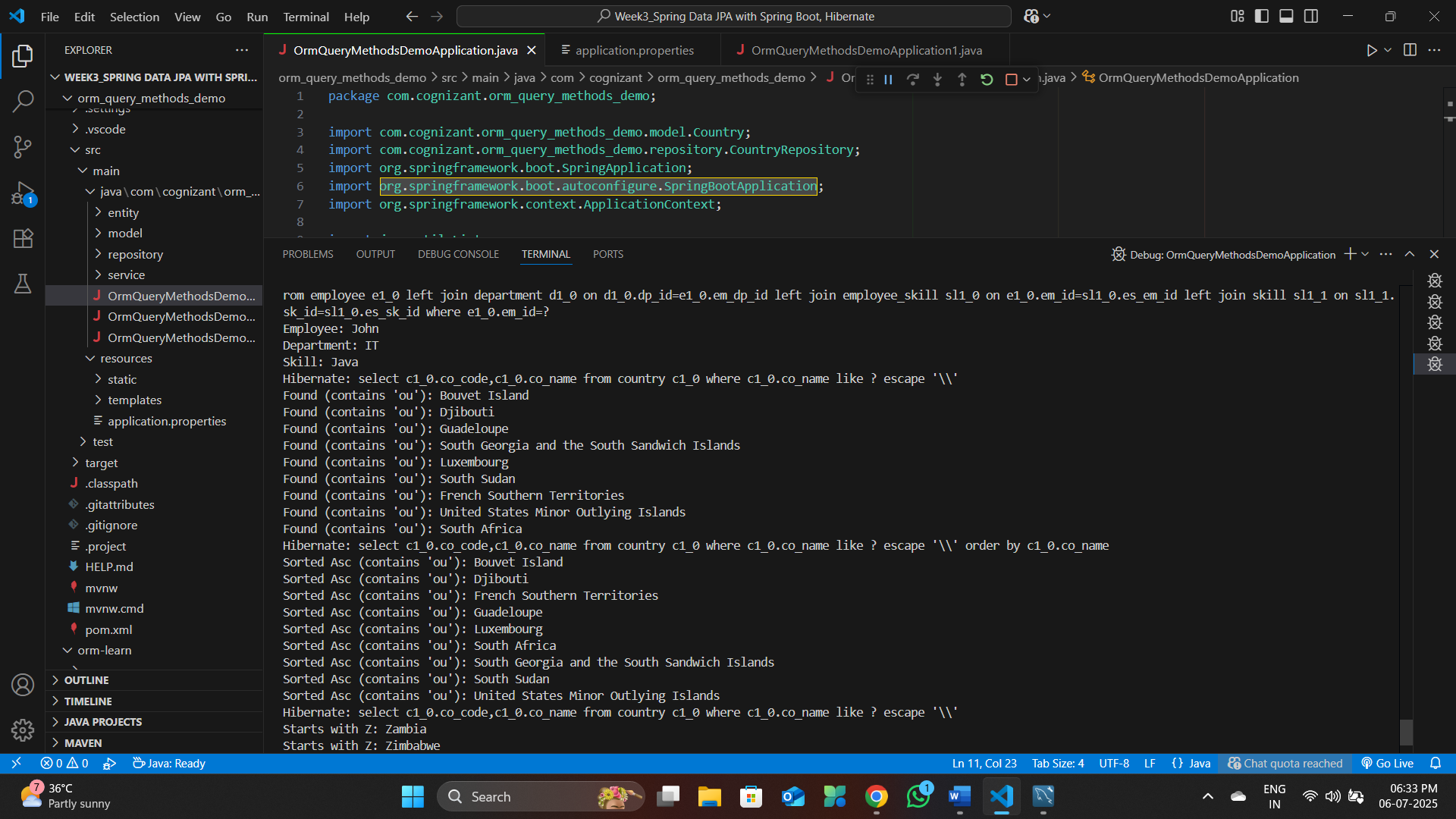
list.forEach(c -> System.out.println("Starts with Z: " + c.getName()));

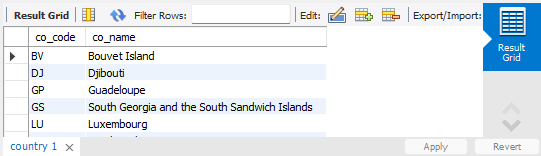
}

}

**Output:**





****

Countries containing 'ou':

South Africa

South Sudan

Bouvet Island

French Southern Territories

Guadeloupe

Luxembourg

United States Minor Outlying Islands

South Georgia and the South Sandwich Islands

Countries containing 'ou' sorted by name:

Bouvet Island

French Southern Territories

Guadeloupe

Luxembourg

South Africa

South Georgia and the South Sandwich Islands

South Sudan

United States Minor Outlying Islands

Countries starting with 'Z':

Zambia

Zimbabwe

**Exercise 7:** **Demonstrate implementation of O/R Mapping**

**MySQL Workbench:**

CREATE DATABASE ormlearn;

USE ormlearn;

CREATE TABLE department (

dp\_id INT PRIMARY KEY AUTO\_INCREMENT,

dp\_name VARCHAR(100)

);

CREATE TABLE employee (

em\_id INT PRIMARY KEY AUTO\_INCREMENT,

em\_name VARCHAR(100),

em\_salary DOUBLE,

em\_permanent BOOLEAN,

em\_date\_of\_birth DATE,

em\_dp\_id INT,

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

);

CREATE TABLE skill (

sk\_id INT PRIMARY KEY AUTO\_INCREMENT,

sk\_name VARCHAR(100)

);

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(em\_id),

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

);

**src/main/resources/application.properties**

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

spring.datasource.username=root

spring.datasource.password=123san@V

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**src/main/java/com/cognizant/orm\_query\_methods\_demo/model/Employee.java**

package com.cognizant.orm\_query\_methods\_demo.model;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.Set;

@Entity

@Table(name = "employee")

public class Employee {

    @Id

    @Column(name = "em\_id")

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    @Column(name = "em\_name")

    private String name;

    @Column(name = "em\_salary")

    private double salary;

    @Column(name = "em\_permanent", nullable = false)

    private boolean permanent;

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

    @Temporal(TemporalType.DATE)

    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;

    // Getters and setters

    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;

    }

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/model/Skill.java**

package com.cognizant.orm\_query\_methods\_demo.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "skill")

public class Skill {

    @Id

    @Column(name = "sk\_id")

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    @Column(name = "sk\_name")

    private String name;

    @ManyToMany(mappedBy = "skillList")

    private Set<Employee> employeeList;

    // Getters and setters

    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;

    }

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/model/Department.java**

package com.cognizant.orm\_query\_methods\_demo.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "department")

public class Department {

    @Id

    @Column(name = "dp\_id")

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    @Column(name = "dp\_name")

    private String name;

    @OneToMany(mappedBy = "department")

    private Set<Employee> employeeList;

    // Getters and setters

    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;

    }

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/repository/**

**EmployeeRepository.java**

package com.cognizant.orm\_query\_methods\_demo.repository;

import com.cognizant.orm\_query\_methods\_demo.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/repository/SkillRepository.java**

package com.cognizant.orm\_query\_methods\_demo.repository;

import com.cognizant.orm\_query\_methods\_demo.model.Skill;

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

public interface SkillRepository extends JpaRepository<Skill, Integer> {

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/repository/**

**DepartmentRepository.java**

package com.cognizant.orm\_query\_methods\_demo.repository;

import com.cognizant.orm\_query\_methods\_demo.model.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/service/EmployeeService.java**

package com.cognizant.orm\_query\_methods\_demo.service;

import com.cognizant.orm\_query\_methods\_demo.model.Employee;

import com.cognizant.orm\_query\_methods\_demo.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

    @Autowired

    private EmployeeRepository employeeRepository;

    public Employee get(int id) {

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

    }

    public void save(Employee employee) {

        employeeRepository.save(employee);

    }

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/service/DepartmentService.java**

package com.cognizant.orm\_query\_methods\_demo.service;

import com.cognizant.orm\_query\_methods\_demo.model.Department;

import com.cognizant.orm\_query\_methods\_demo.repository.DepartmentRepository;

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).orElse(null);

    }

    public void save(Department department) {

        departmentRepository.save(department);

    }

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/service/SkillService.java**

package com.cognizant.orm\_query\_methods\_demo.service;

import com.cognizant.orm\_query\_methods\_demo.model.Skill;

import com.cognizant.orm\_query\_methods\_demo.repository.SkillRepository;

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).orElse(null);

    }

    public void save(Skill skill) {

        skillRepository.save(skill);

    }

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/** **OrmQueryMethodsDemoApplication1.java**

// Main Application Class

package com.cognizant.orm\_query\_methods\_demo;

import com.cognizant.orm\_query\_methods\_demo.model.\*;

import com.cognizant.orm\_query\_methods\_demo.service.\*;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import java.util.\*;

@SpringBootApplication

public class OrmQueryMethodsDemoApplication1 {

    public static void main(String[] args) {

        SpringApplication.run(OrmQueryMethodsDemoApplication.class, args);

    }

    @Bean

    CommandLineRunner run(EmployeeService employeeService, DepartmentService departmentService,

            SkillService skillService) {

        return args -> {

            // Add Department

            Department dept = new Department();

            dept.setName("IT");

            departmentService.save(dept);

            // Add Skill

            Skill skill = new Skill();

            skill.setName("Java");

            skillService.save(skill);

            // Add Employee

            Employee emp = new Employee();

            emp.setName("John");

            emp.setSalary(50000);

            emp.setPermanent(true);

            emp.setDateOfBirth(new Date());

            emp.setDepartment(dept);

            emp.setSkillList(new HashSet<>(Arrays.asList(skill)));

            employeeService.save(emp);

            // Fetch Employee

            Employee fetched = employeeService.get(emp.getId());

            System.out.println("Employee: " + fetched.getName());

            System.out.println("Department: " + fetched.getDepartment().getName());

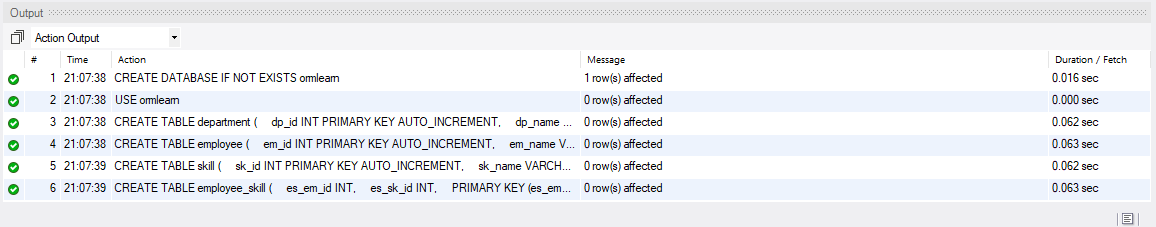
            fetched.getSkillList().forEach(s -> System.out.println("Skill: " + s.getName()));

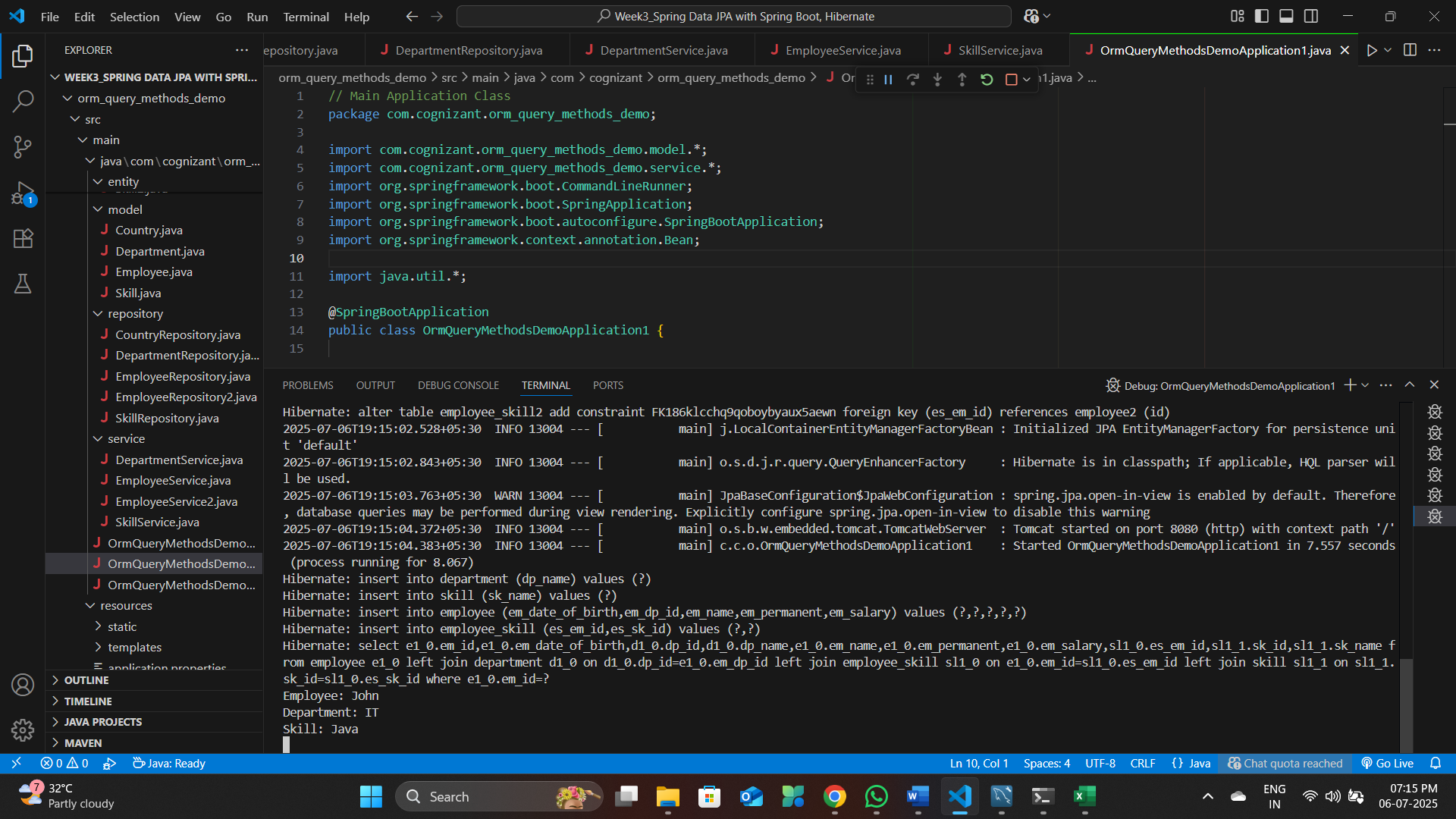
        };

    }

}

**Output:**

****



**Console Output**

Employee Name: John Doe

Department: IT

**Exercise 8:** **Demonstrate writing Hibernate Query Language and Native Query**

**MySQL Workbench:**

CREATE DATABASE ormlearn;

USE ormlearn;

-- Create table for department2

CREATE TABLE department2 (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL

);

-- Create table for skill2

CREATE TABLE skill2 (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL

);

-- Create table for employee2

CREATE TABLE employee2 (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

date\_of\_birth DATE,

salary DOUBLE,

permanent BOOLEAN,

dp\_id INT,

CONSTRAINT fk\_department2 FOREIGN KEY (dp\_id) REFERENCES department2(id)

);

-- Create join table for employee2 and skill2 (Many-to-Many)

CREATE TABLE employee\_skill2 (

es\_em\_id INT,

es\_sk\_id INT,

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

CONSTRAINT fk\_emp2 FOREIGN KEY (es\_em\_id) REFERENCES employee2(id),

CONSTRAINT fk\_skill2 FOREIGN KEY (es\_sk\_id) REFERENCES skill2(id)

);

-- Insert sample data into department2

INSERT INTO department2 (id, name) VALUES

(1, 'Engineering'),

(2, 'HR');

-- Insert sample data into skill2

INSERT INTO skill2 (id, name) VALUES

(1, 'Java'),

(2, 'Spring Boot'),

(3, 'SQL');

-- Insert sample data into employee2

INSERT INTO employee2 (id, name, date\_of\_birth, salary, permanent, dp\_id) VALUES

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

(2, 'Alice', '1992-03-15', 45000, false, 2),

(3, 'Bob', '1988-07-20', 55000, true, 1);

-- Insert sample data into employee\_skill2

INSERT INTO employee\_skill2 (es\_em\_id, es\_sk\_id) VALUES

(1, 1),

(1, 2),

(3, 3);

**src/main/resources/application.properties**

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

spring.datasource.username=root

spring.datasource.password=123san@V

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**src/main/java/com/cognizant/orm\_query\_methods\_demo/entity/Department2.java**

package com.cognizant.orm\_query\_methods\_demo.entity;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Department2 {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    private String name;

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/entity/Employee2.java**

package com.cognizant.orm\_query\_methods\_demo.entity;

import jakarta.persistence.\*;

import lombok.\*;

import java.util.Date;

import java.util.List;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Employee2 {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    private String name;

    @Temporal(TemporalType.DATE)

    private Date dateOfBirth;

    private double salary;

    private boolean permanent;

    @ManyToOne(fetch = FetchType.LAZY)

    @JoinColumn(name = "dp\_id")

    private Department2 department;

    @ManyToMany(fetch = FetchType.LAZY)

    @JoinTable(name = "employee\_skill2", joinColumns = @JoinColumn(name = "es\_em\_id"), inverseJoinColumns = @JoinColumn(name = "es\_sk\_id"))

    private List<Skill2> skillList;

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/entity/Skill2.java**

package com.cognizant.orm\_query\_methods\_demo.entity;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Skill2 {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    private String name;

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/repository/**

**EmployeeRepository2.java**

package com.cognizant.orm\_query\_methods\_demo.repository;

import com.cognizant.orm\_query\_methods\_demo.entity.Employee2;

import org.springframework.data.jpa.repository.\*;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository2 extends JpaRepository<Employee2, Integer> {

    @Query("SELECT e FROM Employee2 e WHERE e.permanent = true")

    List<Employee2> getAllPermanentEmployees();

    @Query("SELECT e FROM Employee2 e LEFT JOIN FETCH e.department d LEFT JOIN FETCH e.skillList WHERE e.permanent = true")

    List<Employee2> getAllPermanentEmployeesWithFetch();

    @Query("SELECT AVG(e.salary) FROM Employee2 e WHERE e.department.id = :id")

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

    @Query(value = "SELECT \* FROM employee2", nativeQuery = true)

    List<Employee2> getAllEmployeesNative();

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/service/EmployeeService2.java**

package com.cognizant.orm\_query\_methods\_demo.service;

import com.cognizant.orm\_query\_methods\_demo.entity.Employee2;

import com.cognizant.orm\_query\_methods\_demo.repository.EmployeeRepository2;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService2 {

    @Autowired

    private EmployeeRepository2 repository;

    public void printAllPermanentEmployees() {

        List<Employee2> list = repository.getAllPermanentEmployeesWithFetch();

        System.out.println("---- HQL Permanent Employees ----");

        list.forEach(e -> {

            System.out.println("Employee: " + e.getName());

            System.out.println("Skills: " + e.getSkillList());

        });

    }

    public void printAllEmployeesNative() {

        List<Employee2> list = repository.getAllEmployeesNative();

        System.out.println("---- Native Query Employees ----");

        list.forEach(e -> System.out.println("Employee: " + e.getName()));

    }

    public double getAverageSalaryByDepartment(int id) {

        return repository.getAverageSalary(id);

    }

}

**src/main/java/com/cognizant/orm\_query\_methods\_demo/ OrmQueryMethodsDemoApplication2.java**

package com.cognizant.orm\_query\_methods\_demo;

import com.cognizant.orm\_query\_methods\_demo.service.EmployeeService2;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmQueryMethodsDemoApplication2 {

    public static void main(String[] args) {

        ApplicationContext context = SpringApplication.run(OrmQueryMethodsDemoApplication2.class, args);

        EmployeeService2 service = context.getBean(EmployeeService2.class);

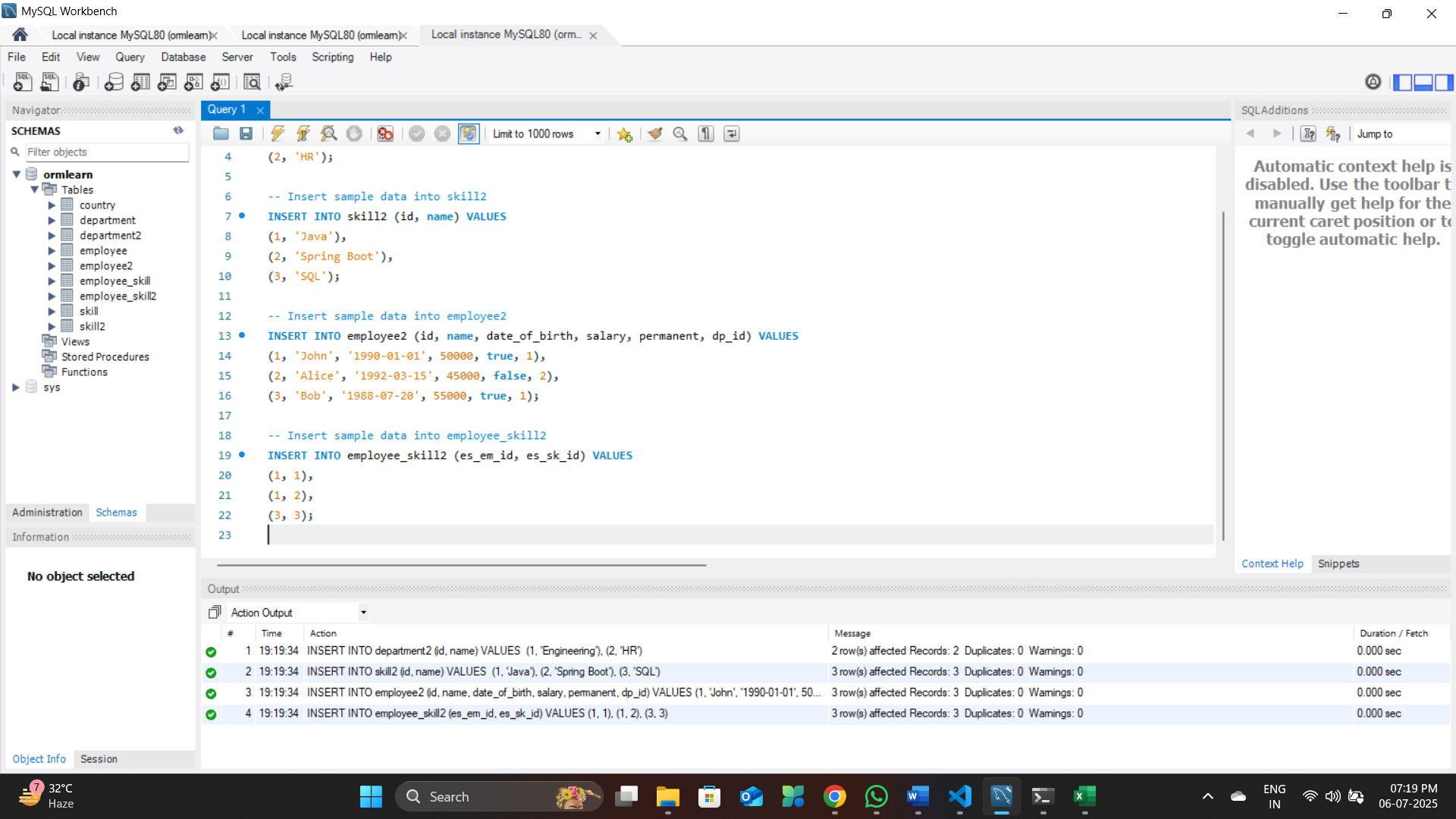
        service.printAllPermanentEmployees();

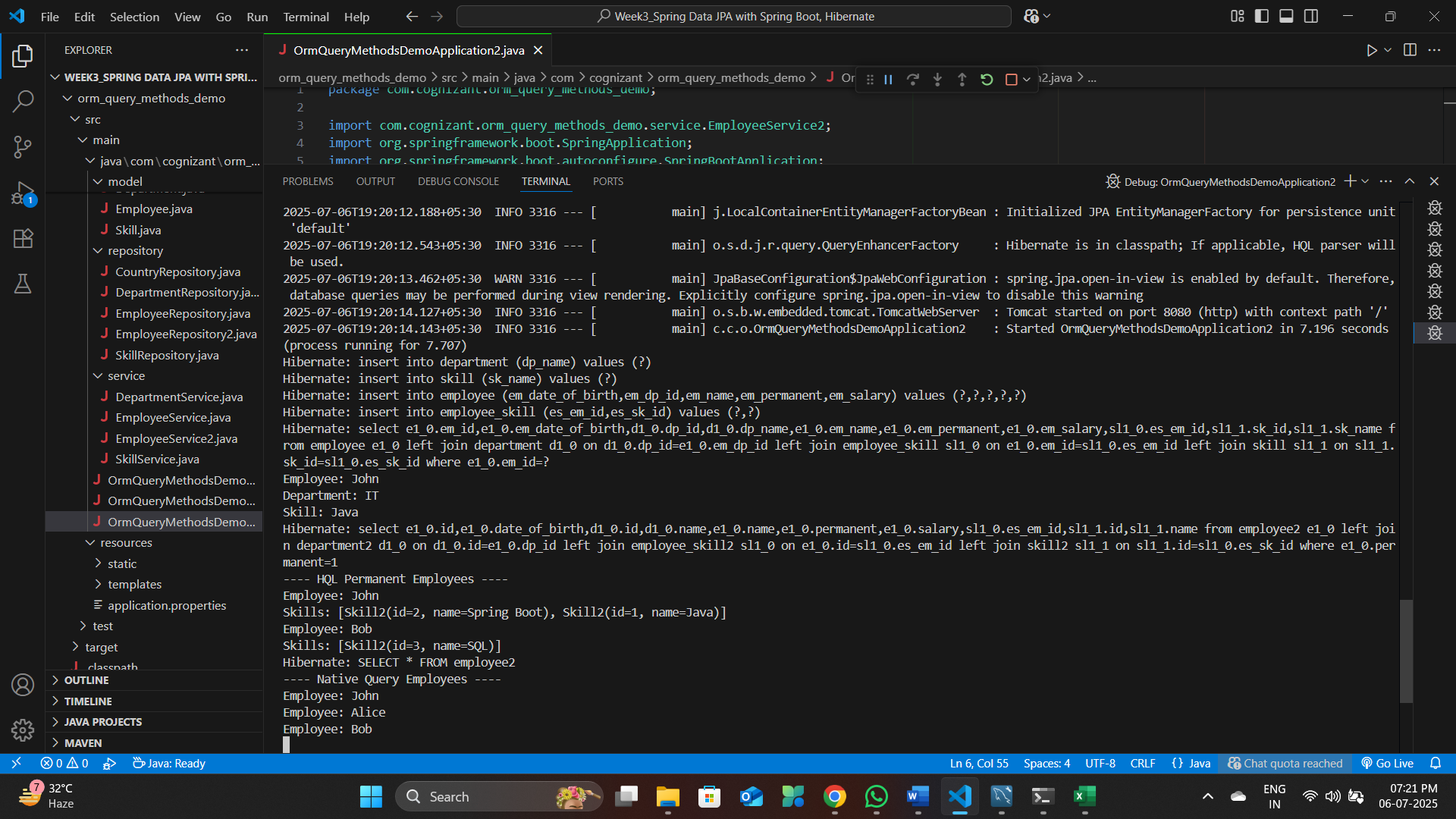
        service.printAllEmployeesNative();

    }

}

**Output:**





Employee: John

Department: IT

Skill: Java

---- HQL Permanent Employees ----

Employee: John

Skills: [Skill2(id=2, name=Spring Boot), Skill2(id=1, name=Java)]

Employee: Bob

Skills: [Skill2(id=3, name=SQL)]

Hibernate: SELECT \* FROM employee2

---- Native Query Employees ----

Employee: John

Employee: Alice

Employee: Bob