Hands on 1:

Spring Data JPA - Quick Example

Answer:

OrmLearnApplication.java:

```
package com.cognizant.ormlearn;

import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.service.CountryService;

@SpringBootApplication
public class OrmLearnApplication {

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

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

    private static void testGetAllCountries() {
        LOGGER.info("Start");
```

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

Country.java:

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

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

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

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

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

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

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

    @Override
```

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

CountryRepository.java:

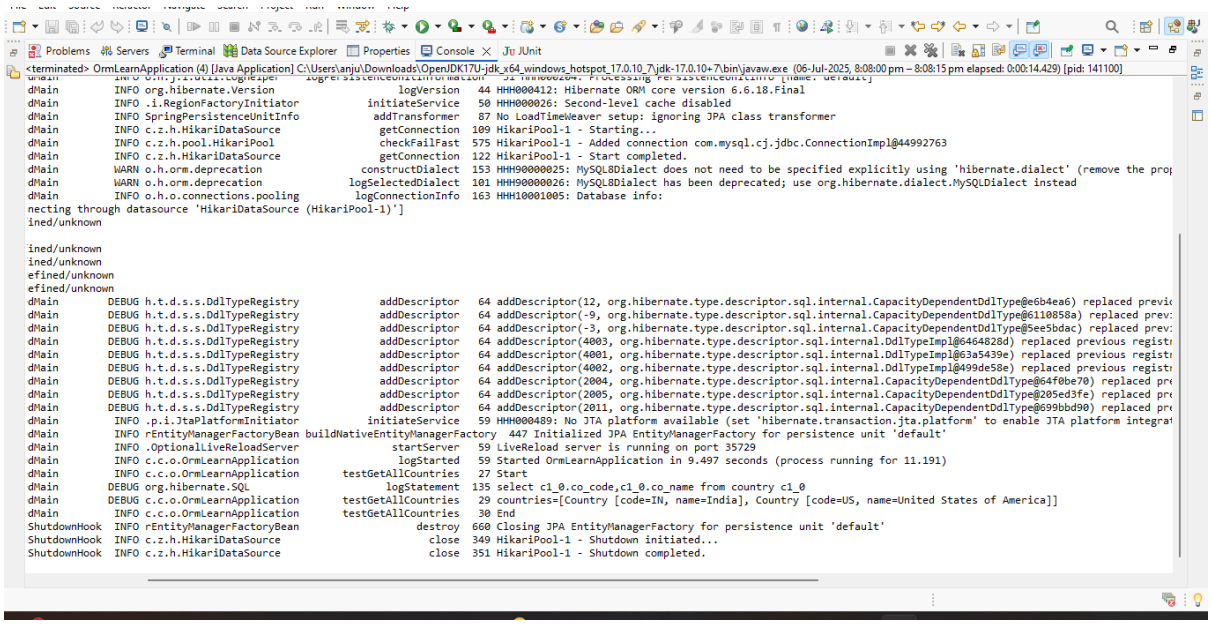
```
package com.cognizant.ormlearn.repository;

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

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

CountryService.java:

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

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

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;
```

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

Output:



Difference between JPA, Hibernate and Spring Data JPA

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

JPA:

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

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

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

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

Example code:

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


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

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

Hibernate

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

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

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

Example code:

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

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

Exercise 1: Configuring a Basic Spring Application

Scenario:

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

Steps:

1. **Set Up a Spring Project:**
    - Create a Maven project named **LibraryManagement**.
    - Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
    - Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
    - Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
    - Create a package **com.library.service** and add a class **BookService**.
    - Create a package **com.library.repository** and add a class **BookRepository**.

4. **Run the Application:**
    - Create a main class to load the Spring context and test the configuration.

Answer:

Pom.xml

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

  <modelVersion>4.0.0</modelVersion>

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

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

</project>
```

BookRepository.java

```
package com.library.repository;
public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Book saved: " + bookName);
```

```
    }
}
```

BookService.java

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

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

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

applicationContext.xml

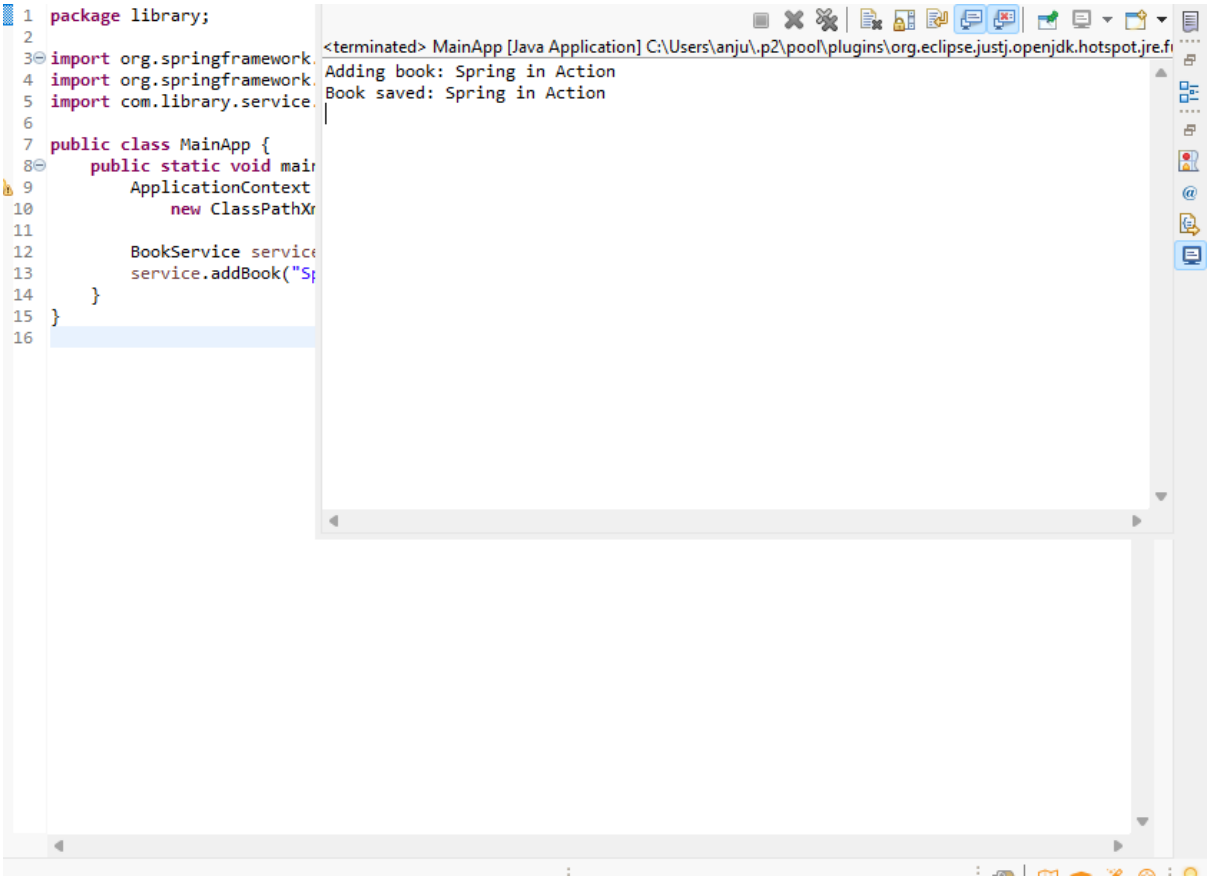
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.library.repository.BookRepository" />

    <bean id="bookService" class="com.library.service.BookService">
```

```
    <property name="bookRepository" ref="bookRepository" />
  </bean>

</beans>
```

MainApp.java

```
package library;

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

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

    BookService service = (BookService) context.getBean("bookService");
    service.addBook("Spring in Action");
  }
}
```

Output:



Exercise 2: Implementing Dependency Injection

Scenario:

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

Steps:

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

Answer:

pom.xml

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

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

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

```
      https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>LibraryManagementApp</artifactId>
<version>1.0</version>

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

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

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

BookRepository.java

```
package com.example.repository;

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

BookService.java

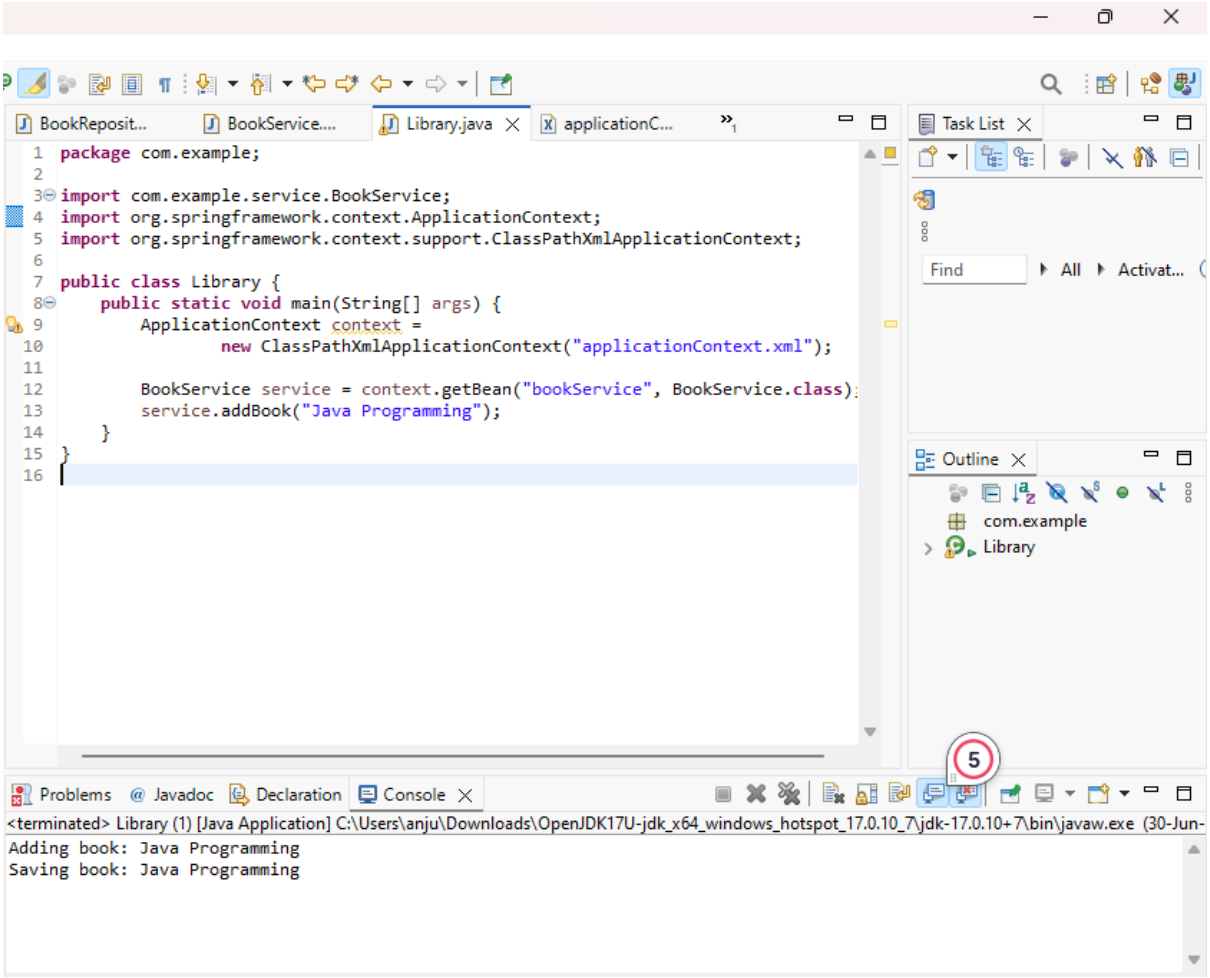
```
package com.example.service;

import com.example.repository.BookRepository;

public class BookService {
```

```
    private BookRepository bookRepository;

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

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

Library.java

```
package com.example;

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

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

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

Output:




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

Exercise 4: Creating and Configuring a Maven Project

Scenario:

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

Steps:

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

Answer:

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

    <modelVersion>4.0.0</modelVersion>

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

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

    <build>
```

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

</project>
```

Exercise 5: Configuring the Spring IoC Container

Scenario:

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

Steps:

1. **Create Spring Configuration File:**
    - Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
    - Define beans for **BookService** and **BookRepository** in the XML file.
2. **Update the BookService Class:**
    - Ensure that the **BookService** class has a setter method for **BookRepository**.
3. **Run the Application:**

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

Answer:

Pom.xml

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

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

<modelVersion>4.0.0</modelVersion>

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

<dependencies>

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

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

<build>
 <plugins>

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

</project>
```

BookService.java

```
package com.example.service;

import com.example.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

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

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

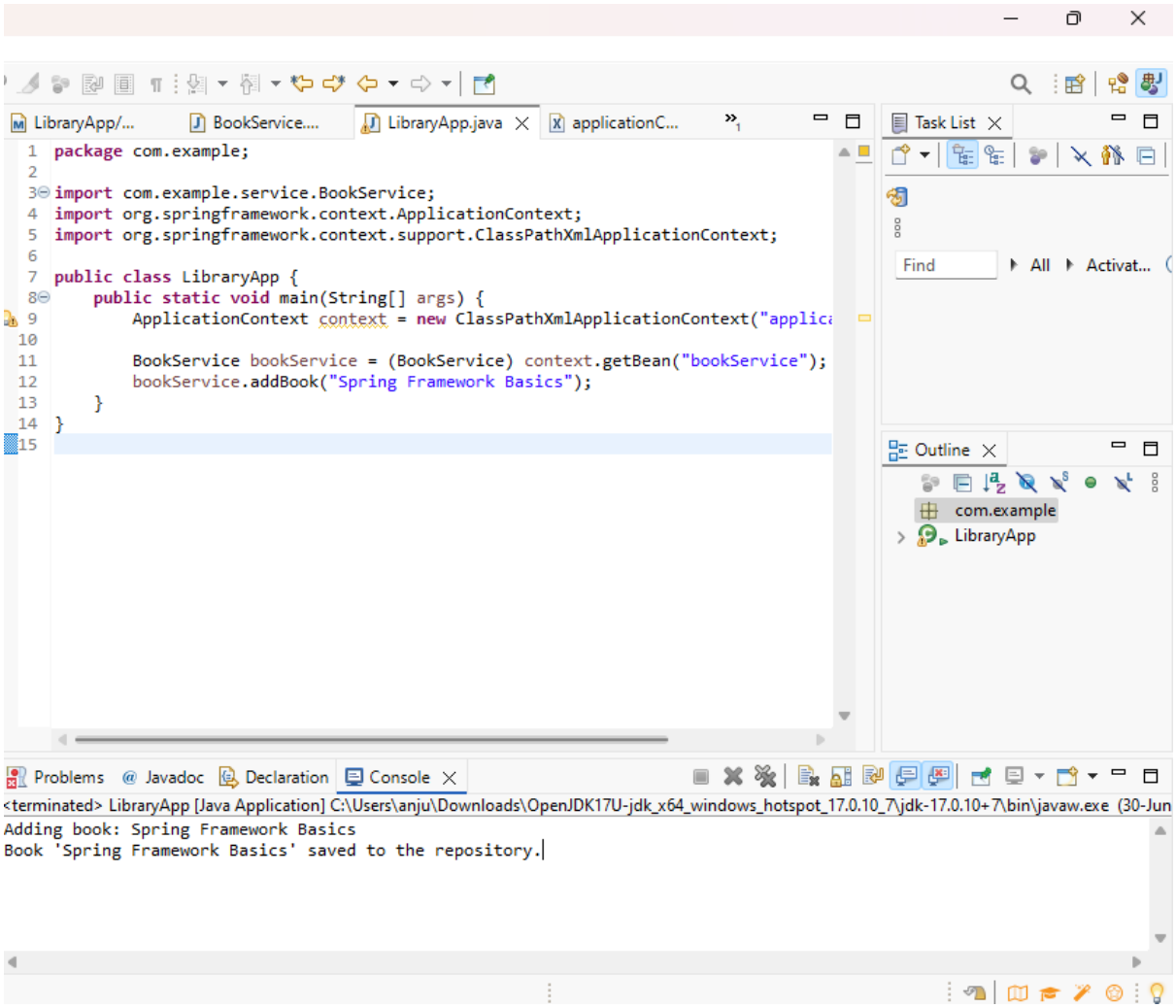
BookRepository.java

```
package com.example.repository;

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

LibraryApp.java

```
package com.example;

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

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

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

Output:




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

Exercise 7: Implementing Constructor and Setter Injection

Scenario:

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

Steps:

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

Answer

pom.xml

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

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

  <dependencies>

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


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

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

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

</project>
```

BookService.java

```
package com.example;

public class BookService {
   private BookRepository bookRepository;

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

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

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

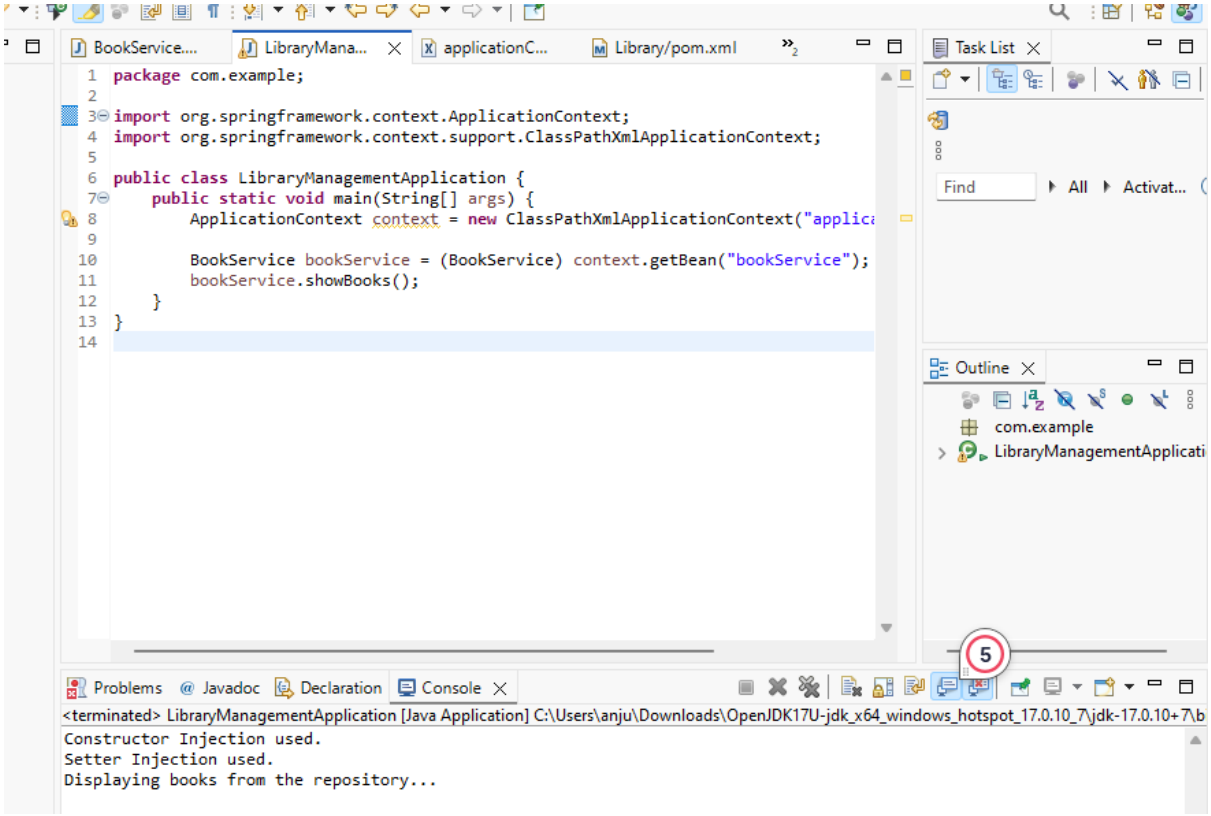
BookRepository.java

```
package com.example;


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

LibraryManagementApplication.java

```
package com.example;


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


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


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

Output:



Exercise 9: Creating a Spring Boot Application

Scenario:

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

Steps:

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

Answer:

Book.java

```
package com.example;

import jakarta.persistence.*;

@Entity
public class Book {

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

    private String title;
    private String author;
    private String isbn;

    // Getters and Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

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

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

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

BookController.java

```java
package com.example;

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

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

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

    @Autowired
    private BookRepository bookRepository;

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

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

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

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

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

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

BookRepository.java

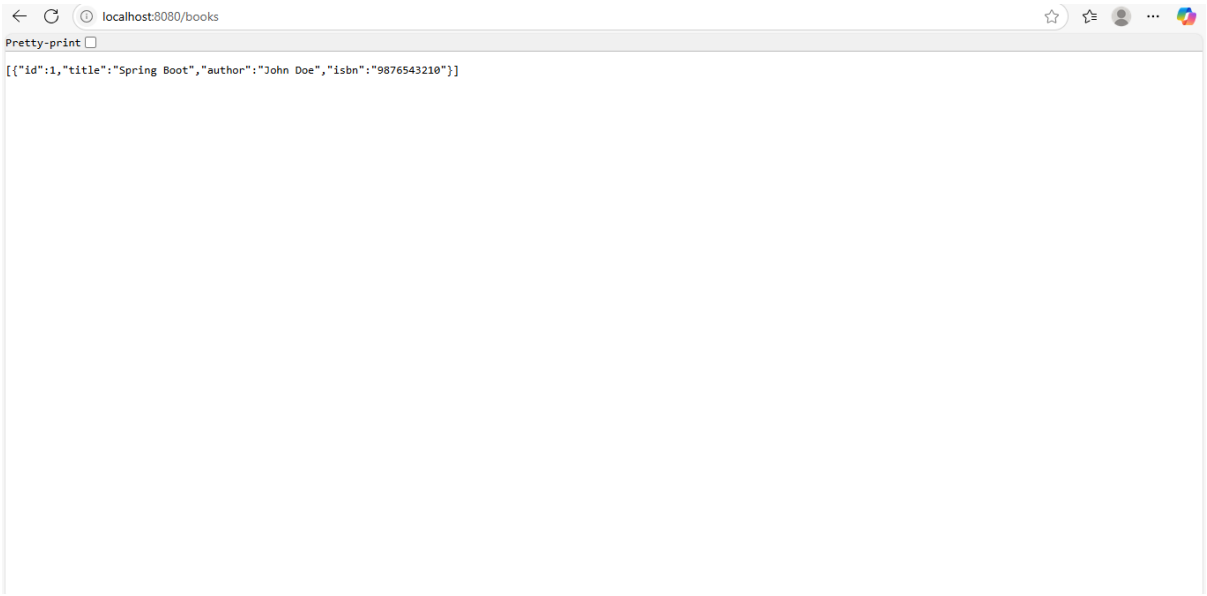
```
package com.example;

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

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

LibraryManagementApplication.java

```
package com.example;

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

@SpringBootApplication
public class LibraryManagementApplication {
    public static void main(String[] args) {
```

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

Output:



Exercise 10: Implement services for managing Country

CountryServiceApplication.java

```
package com.example.country_service;

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

@SpringBootApplication
public class CountryServiceApplication {

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

```
}
```

Country.java

```
import jakarta.persistence.Table;

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

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

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



    public String getCoCode() {
        return coCode;
    }

    public void setCoCode(String coCode) {
        this.coCode = coCode;
    }

    public String getCoName() {
        return coName;
    }

    public void setCoName(String coName) {
        this.coName = coName;
    }
```

```
}
```

CountryController.java

```
package com.example.countryservice;

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

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

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

    @Autowired
    private CountryService countryService;

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

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

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

```
    }

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

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

CountryRepositry.java

```
package com.example.countryservice;

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

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

CountryService.java

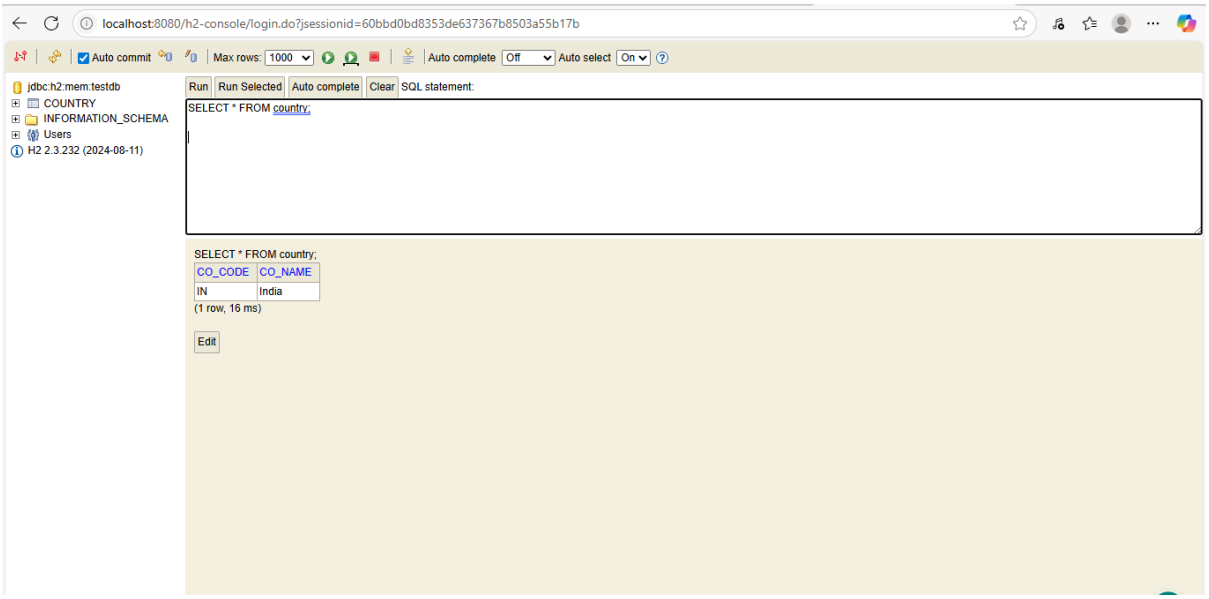
```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;

@Service
public class CountryService {

    @Autowired
```

```
    private CountryRepository countryRepository;


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


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


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


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


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

data.sql

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

Output:



Find a country based on country code

OrmLearnApplication.java

```
package com.cognizant.spring_learn;


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


@SpringBootApplication
public class OrmLearnApplication {


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

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

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

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

Country.java

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

import jakarta.persistence.*;

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

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

```
    private String code;

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

    // Getters & Setters
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

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

CountryNotFoundException.java

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

public class CountryNotFoundException extends Exception {

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

```
    }

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

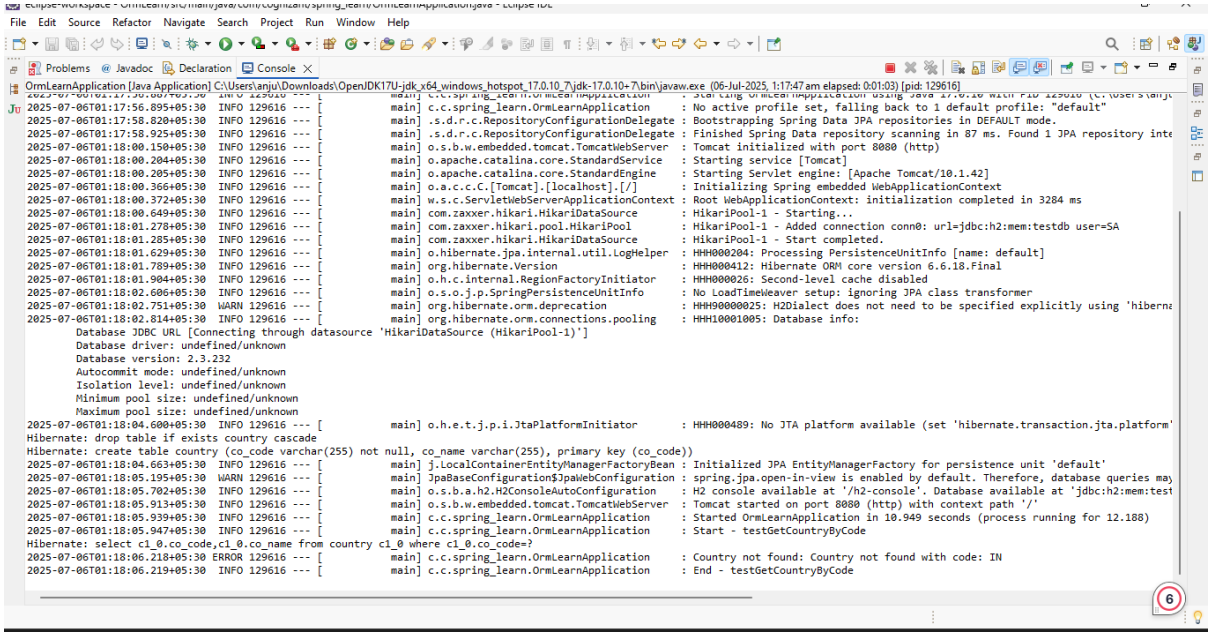
CountryRepository.java

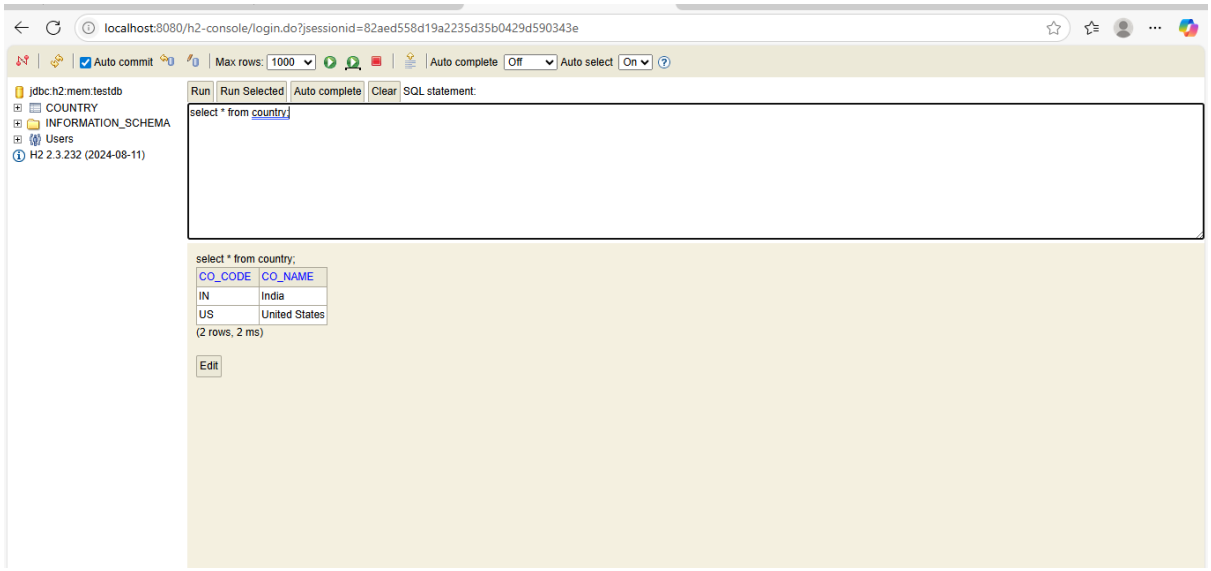
```
package com.cognizant.spring_learn.repository;

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

public interface CountryRepository extends JpaRepository<Country, String> {

}
```

OrmLearnApplicationTests.java

```
package com.cognizant.spring_learn;

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

@SpringBootTest
class OrmLearnApplicationTests {

    @Test
    void contextLoads() {
    }

}
```

Output:





Excerscise:Add a new Country:

Answer:

package com.cognizant.spring_learn.service;

import com.cognizant.spring_learn.model.Country;

import com.cognizant.spring_learn.repository.CountryRepository;

import com.cognizant.spring_learn.service.exception.CountryNotFoundException;

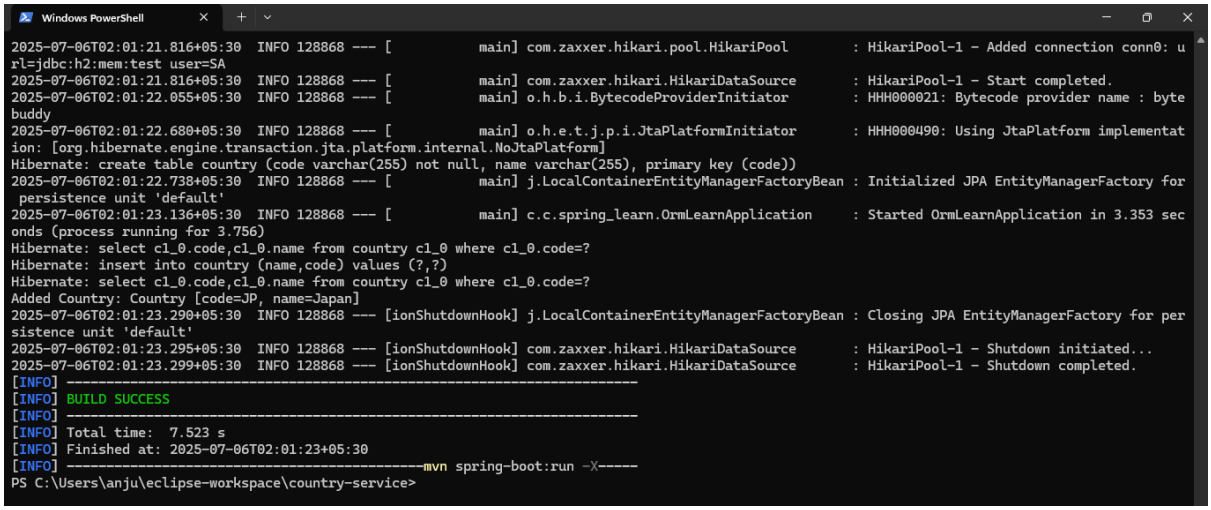
```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Optional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

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

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

package com.cognizant.spring_learn.repository;

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

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

public interface CountryRepository extends JpaRepository<Country, String> {
}
package com.cognizant.spring_learn.model;



import jakarta.persistence.*;

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

    @Id
    private String code;
    private String name;

    // Getters and Setters
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
```

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

package com.cognizant.spring_learn.service.exception;

public class CountryNotFoundException extends Exception {
    public CountryNotFoundException(String message) {
        super(message);
    }
}
import org.springframework.context.ApplicationContext;

@SpringBootApplication
public class OrmLearnApplication {

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

    public static void testAddCountry(ApplicationContext context) {
        CountryService countryService = context.getBean(CountryService.class);

        Country newCountry = new Country();
        newCountry.setCode("JP");
        newCountry.setName("Japan");
```

```
        countryService.addCountry(newCountry);

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

Output:



Exercise:Write queries on country table using Query Methods

Answer:

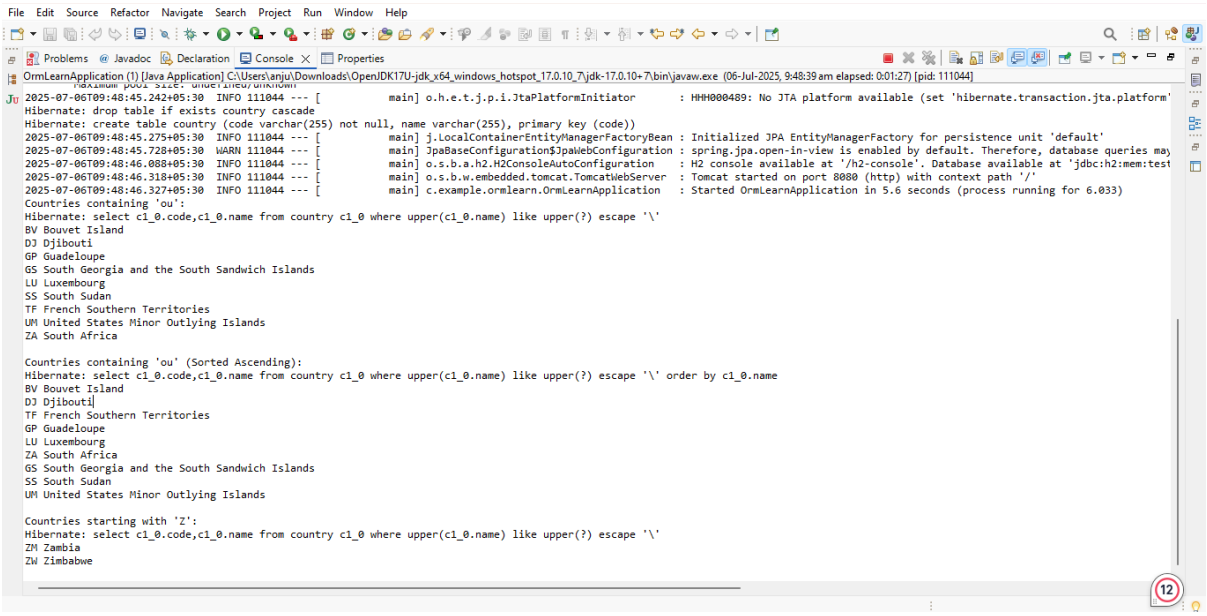
```
package com.example.ormlearn.model;

import jakarta.persistence.*;

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

```
    @Id
    private String code;

    private String name;

    public String getCode() {
        return code;
    }

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

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return code + " " + name;
    }
}
package com.example.ormlearn;

import com.example.ormlearn.model.Country;
import com.example.ormlearn.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
```

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

import java.util.List;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private CountryRepository countryRepository;

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

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

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

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

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

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

import java.util.List;

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

   List<Country> findByNameContainingIgnoreCase(String name);

   List<Country> findByNameContainingIgnoreCaseOrderByNameAsc(String name);



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

Output:



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

Answer:

```
package com.example.ormlearn;


import com.example.ormlearn.model.Stock;

import com.example.ormlearn.repository.StockRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;


import java.math.BigDecimal;

import java.time.LocalDate;

import java.util.List;


@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {


    @Autowired
```

```
    private StockRepository stockRepository;

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

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

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

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

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

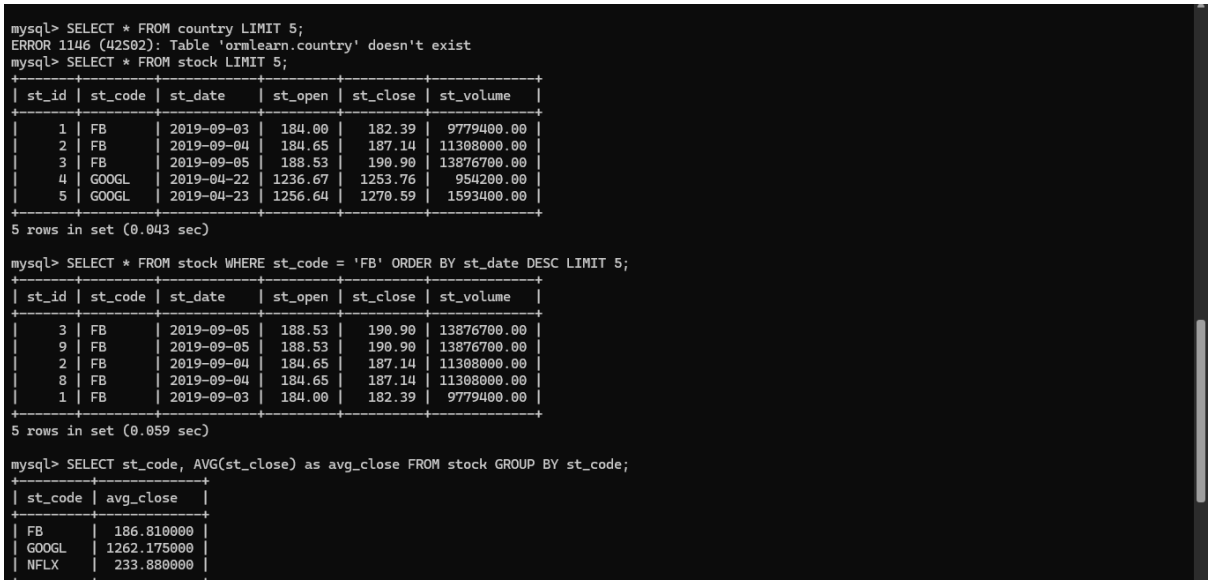
```
package com.example.ormlearn.model;
import javax.persistence.*;
import java.math.BigDecimal;
import java.time.LocalDate;

@Entity
@Table(name = "stock")
public class Stock {

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

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

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

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

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

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

    // Getters and Setters
```

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

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

public String getCode() {
   return code;
}

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

public LocalDate getDate() {
   return date;
}

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

public BigDecimal getOpen() {
   return open;
}

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

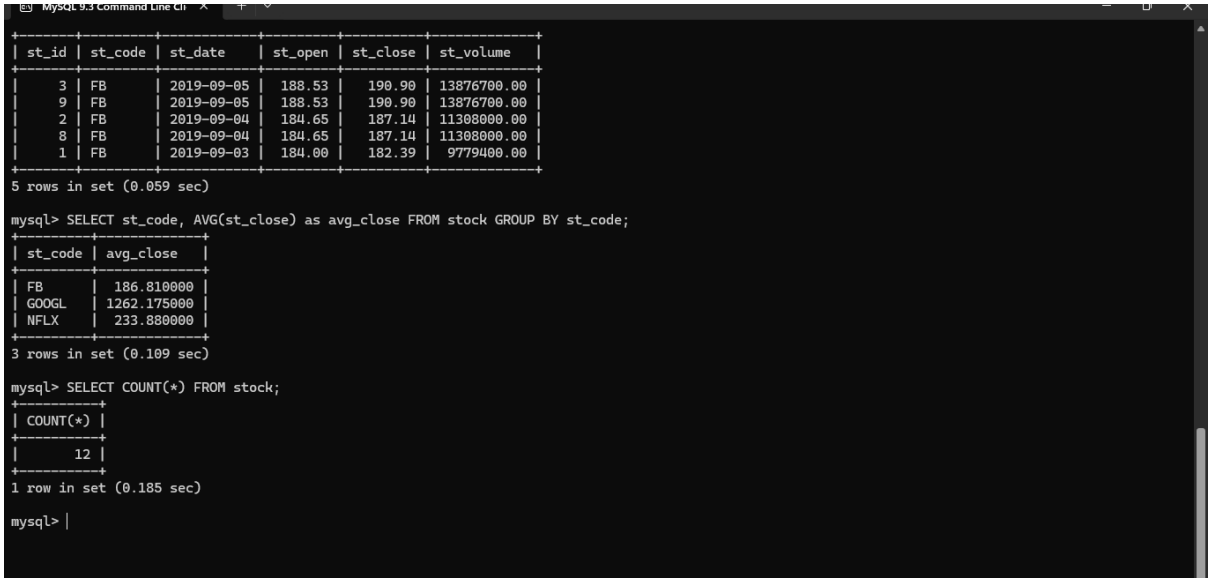
```
    public BigDecimal getClose() {
        return close;
    }

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

    public BigDecimal getVolume() {
        return volume;
    }

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

    @Override
    public String toString() {
        return "Stock{" +
                "code='" + code + '\'' +
                ", date=" + date +
                ", open=" + open +
                ", close=" + close +
                ", volume=" + volume +
                '}';
    }
}
package com.example.ormlearn.repository;

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

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

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

public interface StockRepository extends JpaRepository<Stock, Integer> {
    List<Stock> findByCodeAndDateBetween(String code, LocalDate startDate, LocalDate endDate);
    List<Stock> findByCodeAndCloseGreaterThan(String code, BigDecimal price);

    List<Stock> findTop3ByOrderByVolumeDesc();

    List<Stock> findTop3ByCodeOrderByCloseAsc(String code);
}
```

Output:



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

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

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



Create payroll tables and bean mapping

Answer:

```
package com.cognizant.orm_learn;

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

import java.util.List;

@SpringBootApplication
public class OrmLearnApplication {

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

```
        ConfigurableApplicationContext context = SpringApplication.run(OrmLearnApplication.class,
args);
        EmployeeRepository employeeRepository = context.getBean(EmployeeRepository.class);

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

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

        context.close();
    }
}
package com.cognizant.orm_learn.model;

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

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

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

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

    @OneToMany(mappedBy = "department")
    private Set<Employee> employees;
}
package com.cognizant.orm_learn.model;

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

@Entity
public class Skill {

    @Id
    private int id;

    private String name;


    public int getId() {
        return id;
    }

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

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

```
    }

    public void setName(String name) {
        this.name = name;
    }
}
package com.cognizant.orm_learn.repository;

import com.cognizant.orm_learn.model.Department;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}
package com.cognizant.orm_learn.repository;

import com.cognizant.orm_learn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
}
package com.cognizant.orm_learn.repository;

import com.cognizant.orm_learn.model.Skill;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
```

```
public interface SkillRepository extends JpaRepository<Skill, Integer> {
}
CREATE TABLE IF NOT EXISTS department (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50) NOT NULL
);

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

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

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

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

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

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

VALUES

('Alice Johnson', 75000, TRUE, '1990-03-25', 1),

('Bob Smith', 60000, FALSE, '1992-07-15', 2),

('Charlie Brown', 85000, TRUE, '1985-11-10', 3);

INSERT INTO employee_skill (employee_id, skill_id) VALUES

(1, 1),

(1, 4),

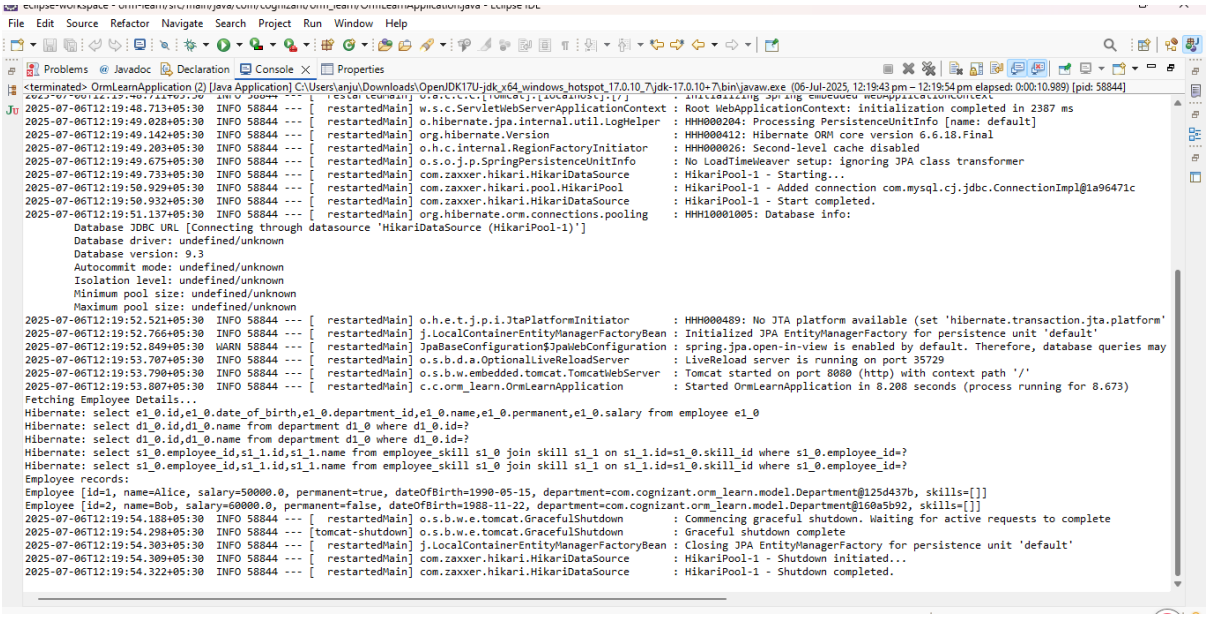
(2, 2),

(2, 4),

(3, 1),

(3, 3);

Output:



Implement many to one relationship between Employee and Department

package com.example.ormlearn;

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

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private DepartmentService departmentService;

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

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

    }

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

```
            System.out.println(employee);
        }
    } else {
        System.out.println("Department not found!");
    }
  }
}
```

Ans

```
package com.example.ormlearn;

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

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

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

  private final EmployeeService employeeService;
```

```
    private final DepartmentRepository departmentRepository;

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

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

    @Override
    public void run(String... args) {
        LOGGER.info("Start");
        Department department = new Department();
        department.setName("IT");
        departmentRepository.save(department);
        Employee employee = new Employee();
        employee.setName("John Doe");
        employee.setSalary(60000.0);
        employee.setPermanent(true);
        employee.setDepartment(department);
        employeeService.save(employee);

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

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

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

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
}
package com.example.ormlearn.repository;

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

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

package com.example.ormlearn;

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

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {
```

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

    private final EmployeeService employeeService;
    private final DepartmentRepository departmentRepository;

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

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

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

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

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

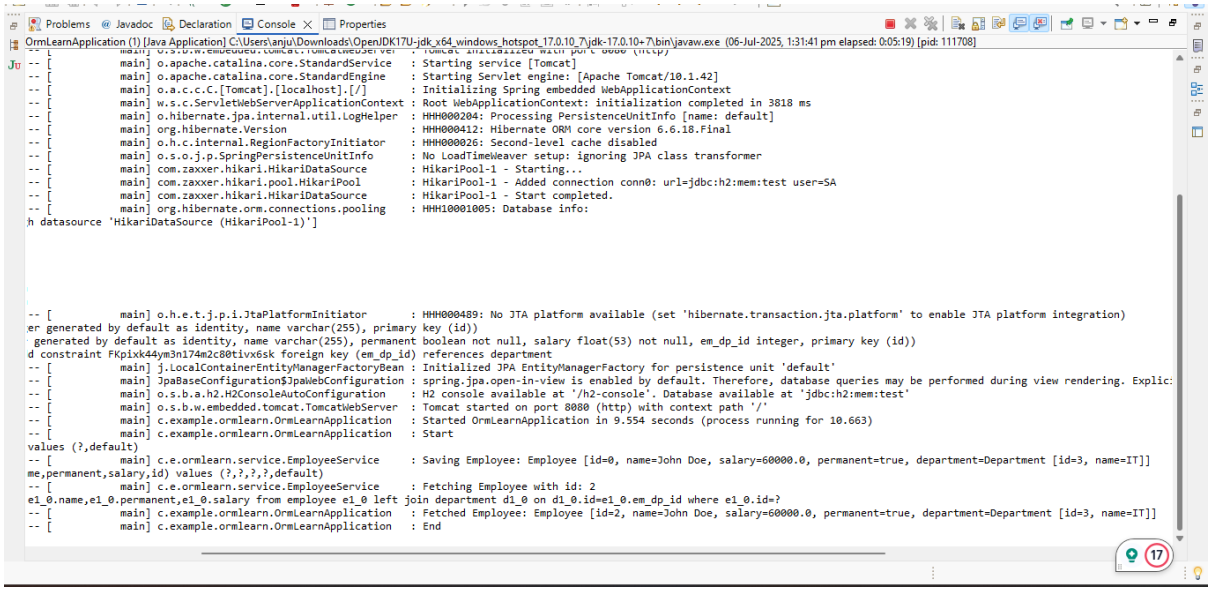
```
        LOGGER.info("End");
    }
}
package com.example.ormlearn.model;

import jakarta.persistence.*;



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

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

    private String name;



    public int getId() {
        return id;
    }

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

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

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

    @Override
    public String toString() {
        return "Department [id=" + id + ", name=" + name + "]";
    }
}
package com.example.ormlearn.service;

import com.example.ormlearn.model.Employee;
import com.example.ormlearn.repository.EmployeeRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class EmployeeService {

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

    private final EmployeeRepository employeeRepository;

    public EmployeeService(EmployeeRepository employeeRepository) {
        this.employeeRepository = employeeRepository;
    }

    @Transactional
```

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

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

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

@Service
public class DepartmentService {

    private final DepartmentRepository departmentRepository;

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

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

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

Output



Implement one to many relationship between Employee and Department

Answer:

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


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


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


    @Id
    private int id;
    private String name;
```

```
@OneToMany(mappedBy = "department", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
private List<Employee> employeeList;

public int getId() {
   return id;
}

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

public String getName() {
   return name;
}

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

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

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

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

```
    }
}

package com.example.ormlearn;

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

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private DepartmentService departmentService;

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

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

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

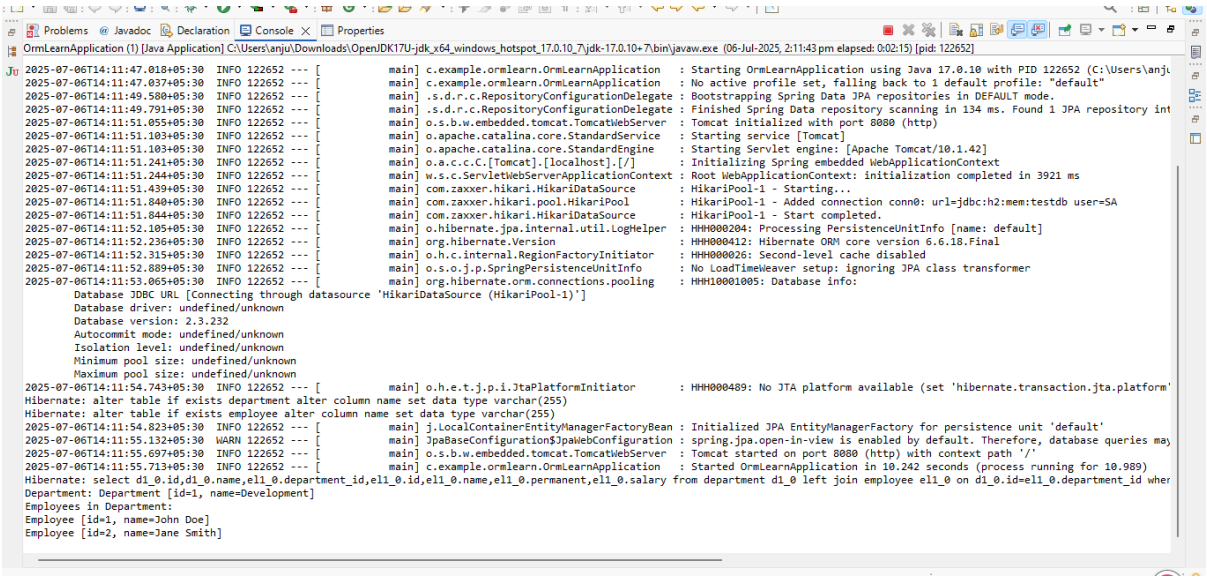
```
            System.out.println("Department: " + department);
            System.out.println("Employees in Department:");
            for (Employee employee : department.getEmployeeList()) {
                System.out.println(employee);
            }
        } else {
            System.out.println("Department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

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}
package com.example.ormlearn.service;

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

import java.util.Optional;

@Service
public class DepartmentService {

    @Autowired
    private DepartmentRepository departmentRepository;
```

```
    public Department get(int id) {
        Optional<Department> result = departmentRepository.findById(id);
        return result.orElse(null);
    }
}
package com.example.ormlearn.model;

import jakarta.persistence.*;



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

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

    private String name;
    private double salary;
    private boolean permanent;

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

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

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

public String getName() {
    return name;
}

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

public double getSalary() {
    return salary;
}

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

public boolean isPermanent() {
    return permanent;
}

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

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

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

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

Output:



Implement many to many relationship between Employee and Skill

```
package com.example.ormlearn;

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

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

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private EmployeeService employeeService;

    @Autowired
    private SkillService skillService;

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

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

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

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

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

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

            LOGGER.debug("Skill added successfully to employee");
            System.out.println("Skill added successfully to employee");
            System.out.println("Updated Employee: " + employee);
            System.out.println("Skills: " + employee.getSkillList());
        } else {
            LOGGER.debug("Employee or Skill not found");
            System.out.println("Employee or Skill not found");
        }
    }
}
package com.example.ormlearn.model;

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

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

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

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

```
private Department department;

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

public int getId() { return id; }
public void setId(int id) { this.id = id; }
public String getName() { return name; }
public void setName(String name) { this.name = name; }
public double getSalary() { return salary; }
public void setSalary(double salary) { this.salary = salary; }
public boolean isPermanent() { return permanent; }
public void setPermanent(boolean permanent) { this.permanent = permanent; }
public Department getDepartment() { return department; }
public void setDepartment(Department department) { this.department = department; }
public Set<Skill> getSkillList() { return skillList; }
public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

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

```
}
```

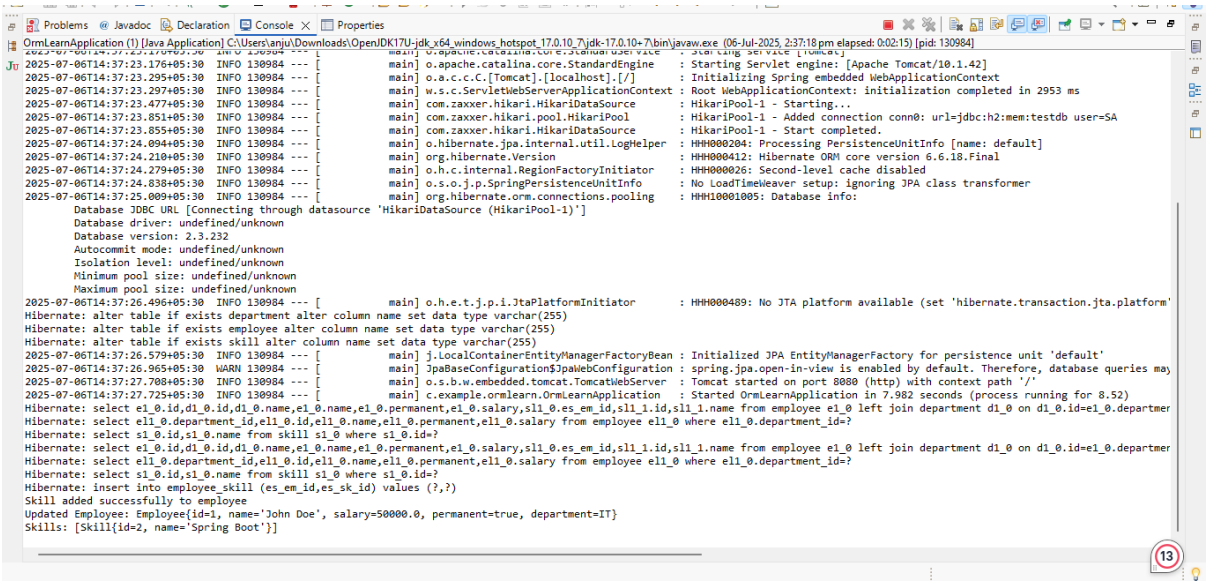
```
package com.example.ormlearn.model;

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

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

    @Id
    private int id;
    private String name;

    @ManyToMany(mappedBy = "skillList")
    private Set<Employee> employeeList = new HashSet<>();

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public Set<Employee> getEmployeeList() { return employeeList; }
    public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }

    @Override
    public String toString() {
        return "Skill{" +
                "id=" + id +
```

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

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

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

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

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

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

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

Output:



Get all permanent employees using HQL

Answer:

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

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

@Service
public class TestEmployeeService {

    @Autowired
    private EmployeeService employeeService;

    @Autowired
    private SkillService skillService;

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

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

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

```
            employeeService.save(employee);
            LOGGER.info("Skill '{}' added successfully to employee '{}'", skill.getName(),
employee.getName());
        } else {
            LOGGER.info("Skill '{}' already exists for employee '{}'", skill.getName(),
employee.getName());
        }
    } else {
        LOGGER.warn("Employee or Skill not found for adding skill.");
    }
  }

  @Transactional
  public void fetchAllPermanentEmployees() {
    LOGGER.info("Fetching all permanent employees...");
    List<Employee> employees = employeeService.getAllPermanentEmployees();

    for (Employee employee : employees) {
        String departmentName = (employee.getDepartment() != null) ?
employee.getDepartment().getName() : "No Department";
        LOGGER.info("Employee: {}, Department: {}, Skills: {}",
            employee.getName(),
            departmentName,
            employee.getSkillList());
    }

    LOGGER.info("Completed fetching permanent employees.");
  }
}
package com.example.ormlearn.model;

import jakarta.persistence.*;
```

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

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

    private String name;

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
package com.example.ormlearn.model;

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

```
import java.util.Set;

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

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

    private String name;

    private boolean permanent;

    private double salary;

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

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

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

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

public String getName() {
    return name;
}

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

public boolean isPermanent() {
    return permanent;
}

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

public double getSalary() {
    return salary;
}

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

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

```
    }

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

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

    public void setSkillList(Set<Skill> skillList) {
        this.skillList = skillList;
    }
}
package com.example.ormlearn.model;

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

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

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

    private String name;

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

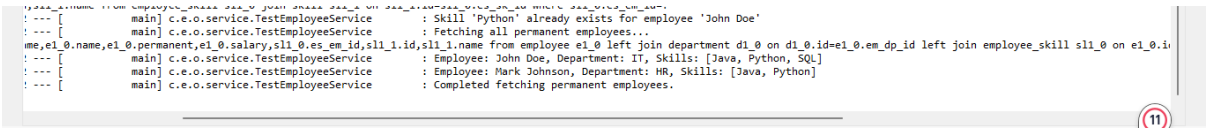
```
}

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

public String getName() {
    return name;
}

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

@Override
public String toString() {
    return name;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Skill)) return false;
    Skill skill = (Skill) o;
    return id == skill.id;
}

@Override
public int hashCode() {
    return Objects.hash(id);
}
```

```
}
```

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

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

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

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

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

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

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

Output:



Fetch Quiz Attemp Details

Answer:

```
package com.example.ormlearn;

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

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

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private AttemptService attemptService;

    @Autowired
    private AttemptRepository attemptRepository;

    @Autowired
    private AttemptQuestionRepository attemptQuestionRepository;
```

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

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

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

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

        attemptQue stionRepository.saveAll(Arrays.asList(q1, q2));

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

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

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

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

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

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

    private String user;

    private Timestamp date;
```

```
@OneToMany(mappedBy = "attempt", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
private List<AttemptQuestion> questions;

public int getId() {
   return id;
}

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

public String getUser() {
   return user;
}

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

public Timestamp getDate() {
   return date;
}

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

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

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

package com.example.ormlearn.model;

import jakarta.persistence.*;

@Entity
public class AttemptOption {

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

    @ManyToOne
    @JoinColumn(name = "attempt_question_id")
    private AttemptQuestion attemptQuestion;

    @ManyToOne
    @JoinColumn(name = "option_id")
    private Option option;

    public Long getId() {
        return id;
    }

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

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

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

    public Option getOption() {
        return option;
    }

    public void setOption(Option option) {
        this.option = option;
    }
}
package com.example.ormlearn.model;

import jakarta.persistence.*;

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

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

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

```
    private Attempt attempt;

    private String questionText;

    public int getId() {
        return id;
    }

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

    public Attempt getAttempt() {
        return attempt;
    }

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

    public String getQuestionText() {
        return questionText;
    }

    public void setQuestionText(String questionText) {
        this.questionText = questionText;
    }
}
package com.example.ormlearn.model;

import jakarta.persistence.*;
```

```
@Entity
public class Option {

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

    private String text;

    private int score;

    @ManyToOne
    @JoinColumn(name = "question_id")
    private Question question;

    public Long getId() {
        return id;
    }

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

    public String getText() {
        return text;
    }

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

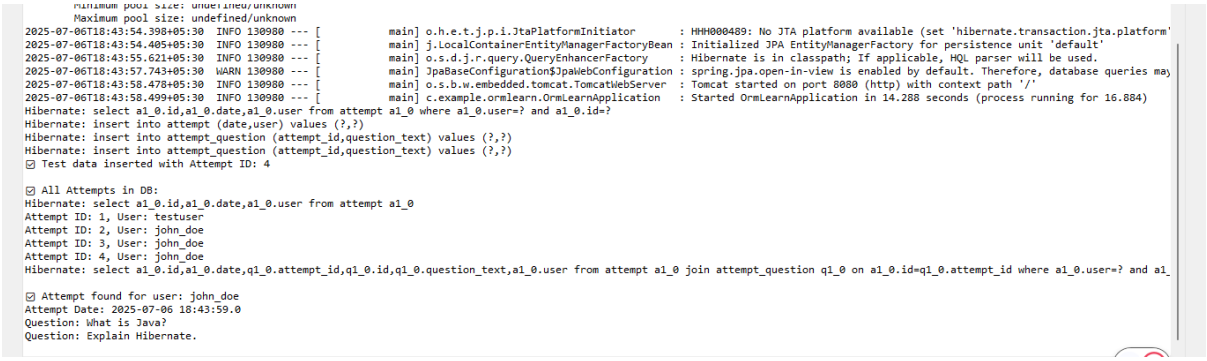
```
    public int getScore() {
        return score;
    }

    public void setScore(int score) {
        this.score = score;
    }

    public Question getQuestion() {
        return question;
    }

    public void setQuestion(Question question) {
        this.question = question;
    }
}

package com.example.ormlearn.model;

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

@Entity
public class Question {

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

    private String text;
```

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

    public Long getId() {
        return id;
    }

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

    public String getText() {
        return text;
    }

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

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

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

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

```
import jakarta.persistence.*;


@Entity
@Table(name = "user")
public class User {


    @Id
    private int id;


    private String name;


    public int getId() {
        return id;
    }


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

Output:



Get average salary using HQL

Answer

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

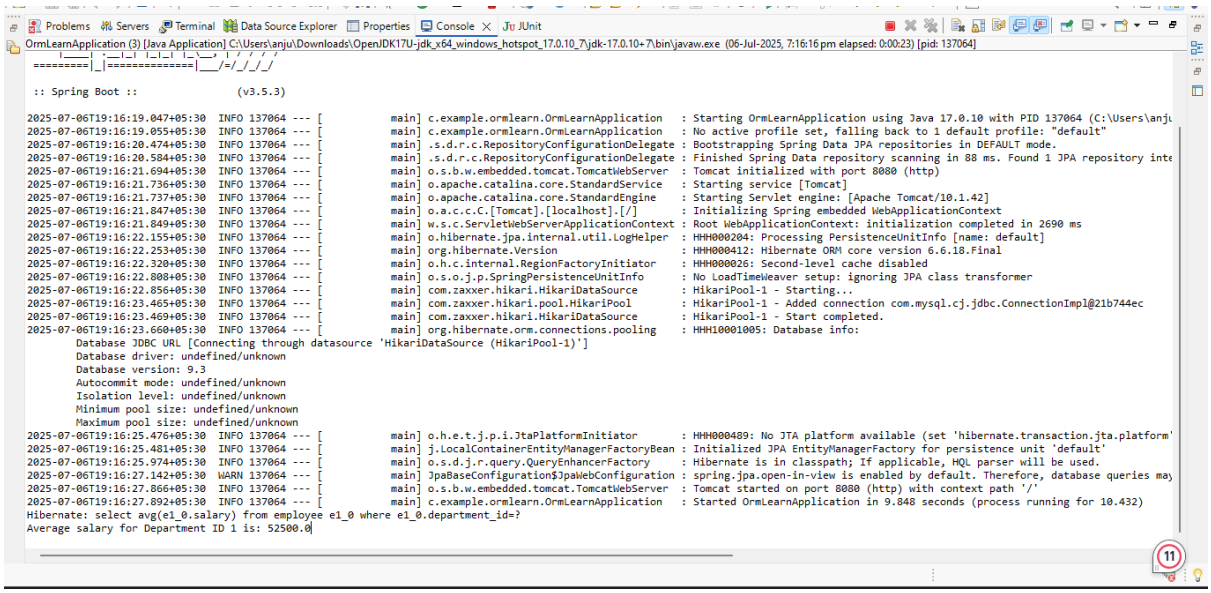
```
import com.example.ormlearn.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private EmployeeService employeeService;

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

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

import jakarta.persistence.*;

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

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

    private String name;
    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
package com.example.ormlearn.model;

import jakarta.persistence.*;

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

    @Id
```

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

private String name;
private double salary;

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

public int getId() {
   return id;
}

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

public String getName() {
   return name;
}

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

public double getSalary() {
   return salary;
}

public void setSalary(double salary) {
```

```
        this.salary = salary;
    }

    public Department getDepartment() {
        return department;
    }

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

package com.example.ormlearn.repository;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

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

Output:



Get all employees using Native Query

Answer:

```
package com.example.ormlearn;


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


import java.util.List;


@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {


    @Autowired
    private EmployeeService employeeService;
```

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

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

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

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

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

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

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

```
@Service
public class EmployeeService {

    @Autowired
    private EmployeeRepository employeeRepository;

    public List<Employee> getAllEmployeesNative() {
        return employeeRepository.getAllEmployeesNative();
    }
}
```

Output: