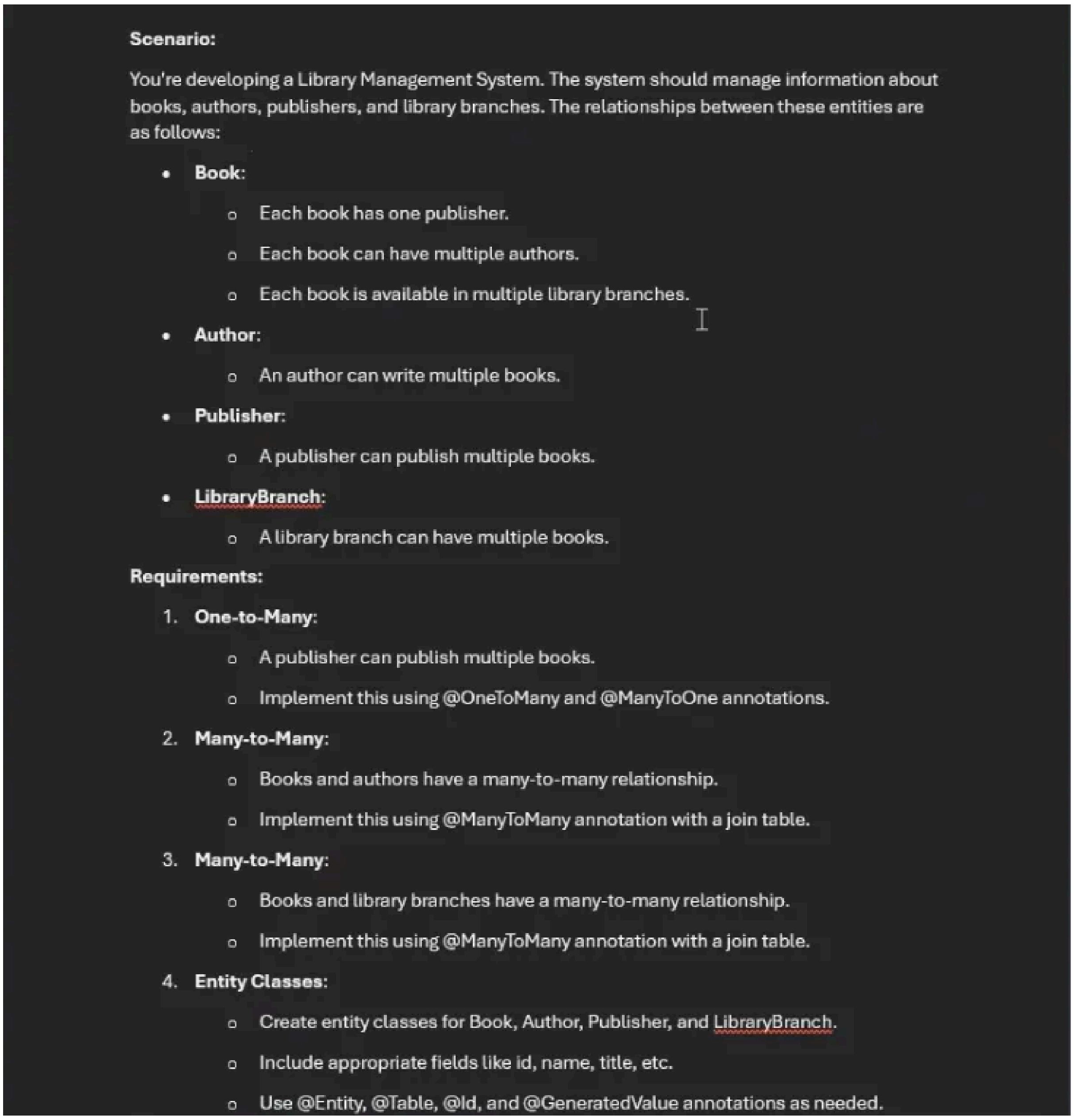
Assignment No. 3

Name - Aditya Pawar

USN - 72233061J

Scenario 1:



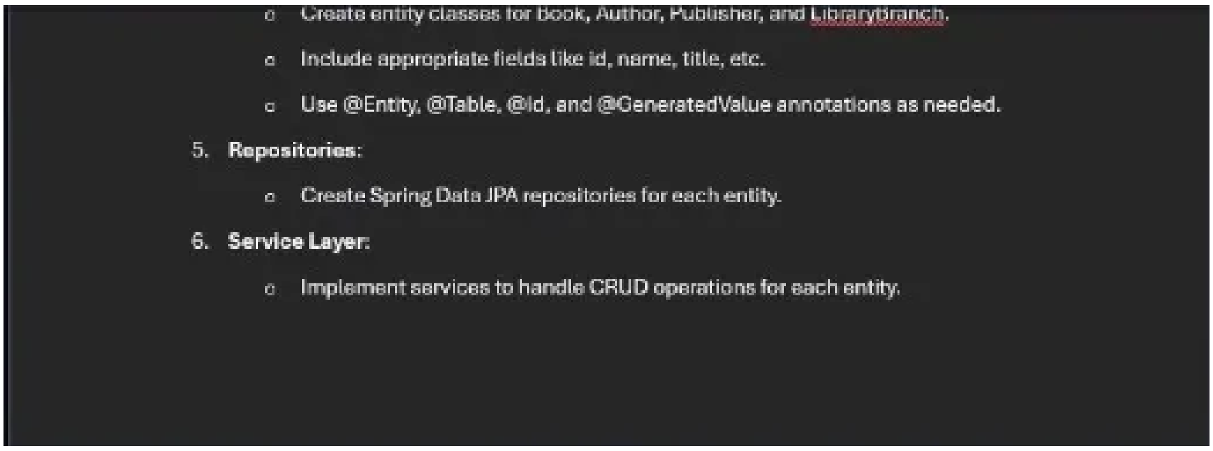
Scenario:

You're developing a Library Management System. The system should manage information about books, authors, publishers, and library branches. The relationships between these entities are as follows:

- **Book**:
  - Each book has one publisher.
  - Each book can have multiple authors.
  - Each book is available in multiple library branches.
- **Author**:
  - An author can write multiple books.
- **Publisher**:
  - A publisher can publish multiple books.
- **LibraryBranch**:
  - A library branch can have multiple books.

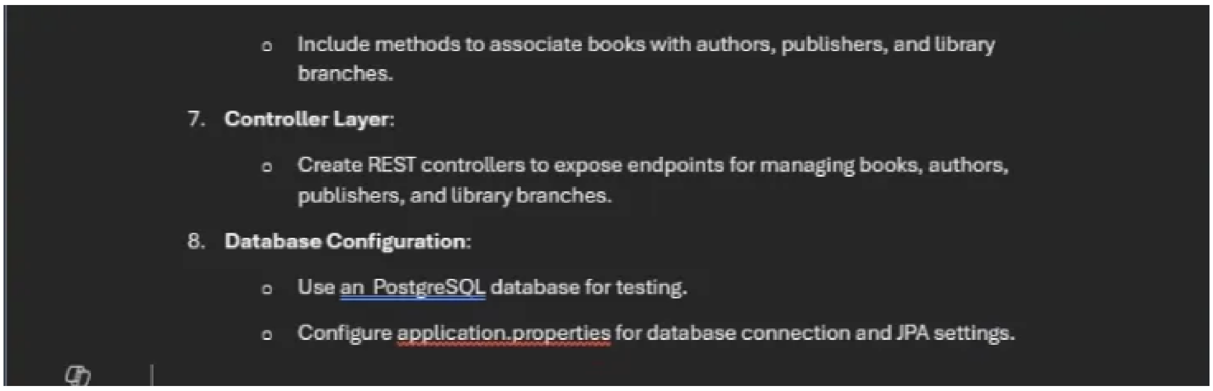
Requirements:

1. **One-to-Many**:
   - A publisher can publish multiple books.
   - Implement this using @OneToMany and @ManyToOne annotations.
2. **Many-to-Many**:
   - Books and authors have a many-to-many relationship.
   - Implement this using @ManyToMany annotation with a join table.
3. **Many-to-Many**:
   - Books and library branches have a many-to-many relationship.
   - Implement this using @ManyToMany annotation with a join table.
4. **Entity Classes**:
   - Create entity classes for Book, Author, Publisher, and LibraryBranch.
   - Include appropriate fields like id, name, title, etc.
   - Use @Entity, @Table, @Id, and @GeneratedValue annotations as needed.



- Create entity classes for Book, Author, Publisher, and LibraryBranch.
- Include appropriate fields like id, name, title, etc.
- Use @Entity, @Table, @Id, and @GeneratedValue annotations as needed.

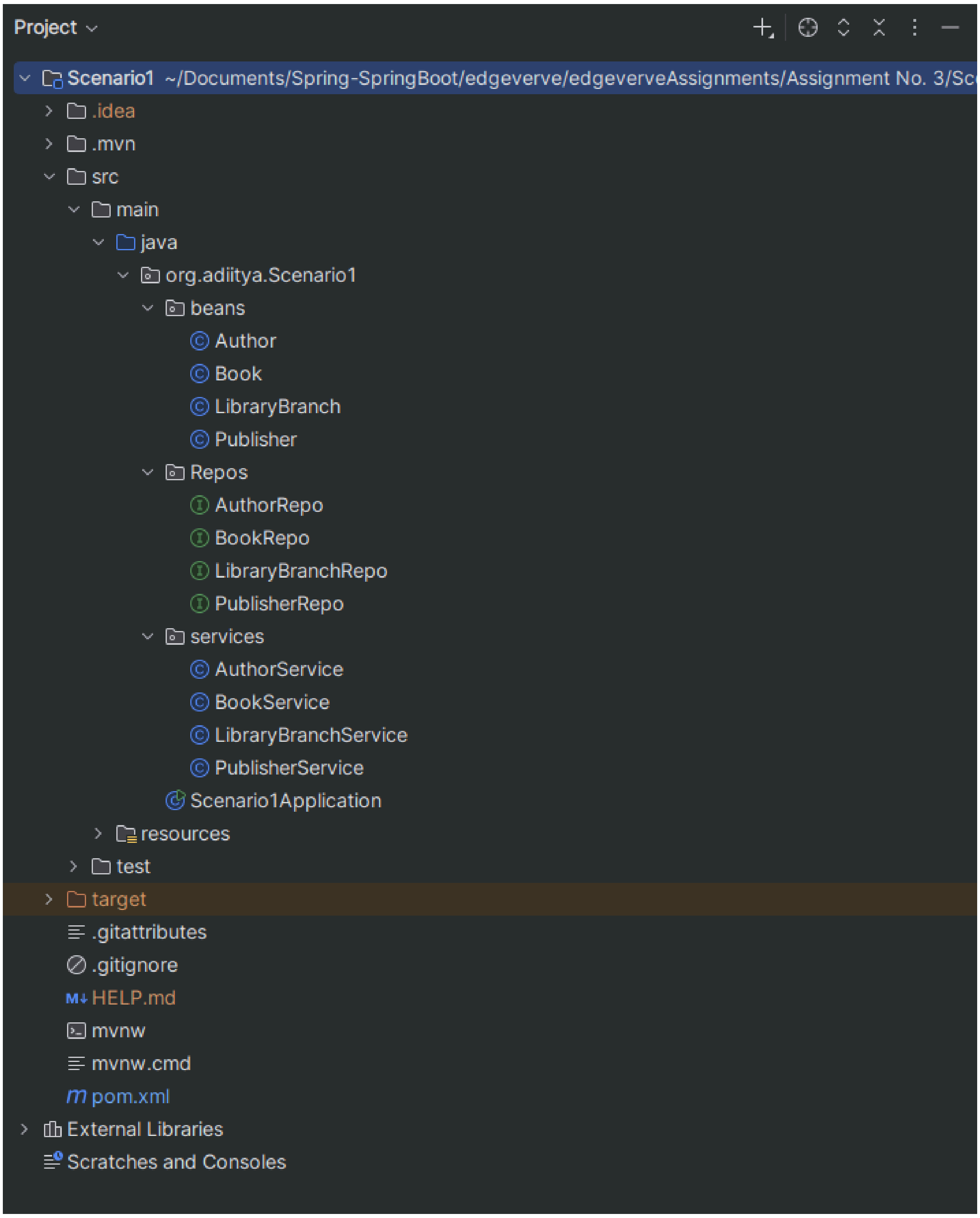
5. **Repositories:**
   - Create Spring Data JPA repositories for each entity.
6. **Service Layer:**
   - Implement services to handle CRUD operations for each entity.



- Include methods to associate books with authors, publishers, and library branches.

7. **Controller Layer:**
   - Create REST controllers to expose endpoints for managing books, authors, publishers, and library branches.
8. **Database Configuration:**
   - Use an PostgreSQL database for testing.
   - Configure application.properties for database connection and JPA settings.

Folder Structure:


```
Project
Scenario1  ~/Documents/Spring-SpringBoot/edgeverve/edgeverveAssignments/Assignment No. 3/Sc
  .idea
  .mvn
  src
    main
      java
        org.adiitya.Scenario1
          beans
            Author
            Book
            LibraryBranch
            Publisher
          Repos
            AuthorRepo
            BookRepo
            LibraryBranchRepo
            PublisherRepo
          services
            AuthorService
            BookService
            LibraryBranchService
            PublisherService
          Scenario1Application
      resources
    test
  target
  .gitattributes
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

POM.XML:

```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://maver
```

```xml
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>3.5.0</version>
   <relativePath/> <!-- lookup parent from repository -->
</parent>

<groupId>com.aditya</groupId>
<artifactId>Scenario1</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>Scenario1</name>
<description>Library Mapping System using Spring Boot</description>

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

<dependencies>
   <!-- Spring Boot Starters -->
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
   </dependency>

   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
   </dependency>

   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>

   <!-- Lombok -->
```

```xml
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.32</version>
      <scope>provided</scope>
    </dependency>

    <!-- PostgreSQL Driver -->
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <scope>runtime</scope>
    </dependency>

    <!-- DevTools (hot reload) -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>

    <!-- For Testing -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <!-- Maven Compiler Plugin -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.11.0</version>
        <configuration>
```

```
                <source>17</source>
                <target>17</target>
                <annotationProcessorPaths>
                    <path>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <version>1.18.32</version>
                    </path>
                </annotationProcessorPaths>
            </configuration>
        </plugin>

        <!-- Spring Boot Maven Plugin for packaging as executable JAR →
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>3.2.5</version>
            <executions>
                <execution>
                    <goals>
                        <goal>repackage</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>

</project>
```

Application.properties:

```
spring.application.name=Scenario1
# ===============================
# DATABASE CONFIGURATION
# ===============================
spring.datasource.url=jdbc:postgresql://localhost:5432/assignment3
spring.datasource.username=postgres
spring.datasource.password=root

# Load database Driver
#spring.datasource.driver-class-name=org.postgresql.Driver

# ================================
# JPA / HIBERNATE CONFIGURATION
# ================================
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=false

# Naming strategy (CamelCase → snake_case table/column names)
#spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.nar
.PhysicalNamingStrategyStandardImpl

# ================================
# SERVER SETTINGS
# ================================
server.port=8080

spring.data.jpa.repositories.bootstrap-mode=default
spring.data.defer-datasource-initalization=true
```

Author.java

```
package org.adiitya.Scenario1.beans;

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

@Entity
public class Author {

    @Id
```

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

private String name;

@ManyToMany(mappedBy = "authors", cascade = CascadeType.ALL)
private List<Book> bookList;

// constructors
public Author() {
}

public Author(String name) {
    this.name = name;
}

// 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 List<Book> getBookList() {
    return bookList;
}
public void setBookList(List<Book> bookList) {
    this.bookList = bookList;
}
```

```
    // toString method

    @Override
    public String toString() {
        return "Author{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
}
```

Book.java

```
package org.adiitya.Scenario1.beans;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import java.util.List;

@Entity
public class Book {

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

    @NotBlank(message = "Book title is important")
    private String title;

    // Many-to-One relationship → Publisher
    @ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinColumn(name = "publisher_id")
    private Publisher publisher;

    // Many-to-Many relationship ⟷ Author
    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinTable(
            name = "book_author",
            joinColumns = @JoinColumn(name = "book_id"),
            inverseJoinColumns = @JoinColumn(name = "author_id")
    )
    private List<Author> authors;

    // Many-to-Many relationship ⟷ LibraryBranch
    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(
            name = "book_branch",
            joinColumns = @JoinColumn(name = "book_id"),
            inverseJoinColumns = @JoinColumn(name = "branch_id")
    )
    private List<LibraryBranch> libraryBranches;

    public Book(){}

    public Book(String title) {
        this.title = title;
    }

    // Getters & setters

    public int getId() {
        return id;
    }

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

    public List<LibraryBranch> getLibraryBranches() {
        return libraryBranches;
    }

    public void setLibraryBranches(List<LibraryBranch> libraryBranches) {
        this.libraryBranches = libraryBranches;
    }
```

```
    public List<Author> getAuthors() {
        return authors;
    }

    public void setAuthors(List<Author> authors) {
        this.authors = authors;
    }

    public Publisher getPublisher() {
        return publisher;
    }

    public void setPublisher(Publisher publisher) {
        this.publisher = publisher;
    }

    public String getTitle() {
        return title;
    }

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

    // ToString method
    @Override
    public String toString() {
        return "Book{" +
                "id=" + id +
                ", title='" + title + '\'' +
                '}';
    }
}
```

LibraryBranch.java

```
package org.adiitya.Scenario1.beans;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;

import java.util.List;

@Entity
public class LibraryBranch {

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

    @NotBlank(message = "Branch name is mandatory")
    private String name;

    @NotBlank(message = "Branch location is mandatory")
    private String location;

    @ManyToMany(mappedBy = "libraryBranches", cascade = CascadeType.A
    private List<Book> books;

    // constructors
    public LibraryBranch(){}

    public LibraryBranch(String name, String location) {
        this.name = name;
        this.location = location;
    }

    // Getters and setters

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

    public List<Book> getBooks() {
        return books;
    }
    public void setBooks(List<Book> books) {
        this.books = books;
    }

    public String getLocation() {
        return location;
    }
    public void setLocation(String location) {
        this.location = location;
    }

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

    @Override
    public String toString() {
        return "LibraryBranch{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", location='" + location + '\'' +
                '}';
    }
}
```

Publisher.java

```
package org.adiitya.Scenario1.beans;
```

```
import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import java.util.List;

@Entity
public class Publisher {

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

    @NotBlank(message = "Publisher name is required")
    private String name;

    @OneToMany(mappedBy = "publisher", cascade = CascadeType.ALL)
    private List<Book> books;

    public Publisher() {
    }

    public Publisher(String name) {
        this.name = name;
    }

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

    public List<Book> getBooks() {
        return books;
    }
    public void setBooks(List<Book> books) {
        this.books = books;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Publisher{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
}
```

Repos:

```
//*****************************************************************

package org.adiitya.Scenario1.Repos;

import org.adiitya.Scenario1.beans.Author;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface AuthorRepo extends JpaRepository<Author, Integer> {
}

//*****************************************************************

package org.adiitya.Scenario1.Repos;

import org.adiitya.Scenario1.beans.Book;
import org.adiitya.Scenario1.beans.Publisher;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface PublisherRepo extends JpaRepository<Publisher, Integer> {
}

//*******************************************************************

package org.adiitya.Scenario1.Repos;

import org.adiitya.Scenario1.beans.LibraryBranch;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface LibraryBranchRepo extends JpaRepository<LibraryBranch, In
}

//*******************************************************************

package org.adiitya.Scenario1.Repos;

import org.adiitya.Scenario1.beans.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface BookRepo extends JpaRepository<Book, Integer> {
}

//*******************************************************************
```

AuthorService.java

}

```
package org.adiitya.Scenario1.services;

import org.adiitya.Scenario1.Repos.AuthorRepo;
import org.adiitya.Scenario1.beans.Author;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class AuthorService {

    @Autowired
    private AuthorRepo authorRepo;

    public void addAuthor(Author author) {
        authorRepo.save(author);
    }

    public Author getAuthorById(int id) {
        return authorRepo.findById(id).orElse(null);
    }

    public List<Author> getAllAuthors() {
        return authorRepo.findAll();
    }

    public void updateAuthor(Author author) {
        authorRepo.save(author);
    }

    public void deleteAuthor(int id) {
        authorRepo.deleteById(id);
    }
}
```

BookService.java

}

```
package org.adiitya.Scenario1.services;

import org.adiitya.Scenario1.Repos.BookRepo;
import org.adiitya.Scenario1.beans.Book;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class BookService {

    @Autowired
    private BookRepo bookRepo;

    public void addBook(Book book){
        bookRepo.save(book);
    }

    public Book getBookById(int id){
        return bookRepo.findById(id).orElse(null);
    }

    public List<Book> getAllBooks() {
        return bookRepo.findAll();
    }

    public void updateBook(Book book){
        bookRepo.save(book);
    }

    public void deleteBook(int id){
        bookRepo.deleteById(id);
    }
}
```

LibraryService.java

```
package org.adiitya.Scenario1.services;

import org.adiitya.Scenario1.Repos.LibraryBranchRepo;
import org.adiitya.Scenario1.beans.LibraryBranch;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class LibraryBranchService {

    @Autowired
    private LibraryBranchRepo branchRepo;

    public void addBranch(LibraryBranch branch){
        branchRepo.save(branch);
    }

    public LibraryBranch getBranchById(int id){
        return branchRepo.findById(id).orElse(null);
    }

    public List<LibraryBranch> getAllBranches(){
        return branchRepo.findAll();
    }
}
```

PublisherService.java

```
package org.adiitya.Scenario1.services;

import org.adiitya.Scenario1.Repos.PublisherRepo;
import org.adiitya.Scenario1.beans.Publisher;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class PublisherService {

    @Autowired
    private PublisherRepo publisherRepo;

    public void addPublisher(Publisher publisher){
        publisherRepo.save(publisher);
    }

    public List<Publisher> getAllPublishers() {
        return publisherRepo.findAll();
    }

    public Publisher getPublisherById(int id){
        return publisherRepo.findById(id).orElse(null);
    }

    public void updatePublisher(Publisher publisher){
        publisherRepo.save(publisher);
    }

    public void deletePublisher(int id){
        publisherRepo.deleteById(id);
    }
}
```

Scenario1Application.java

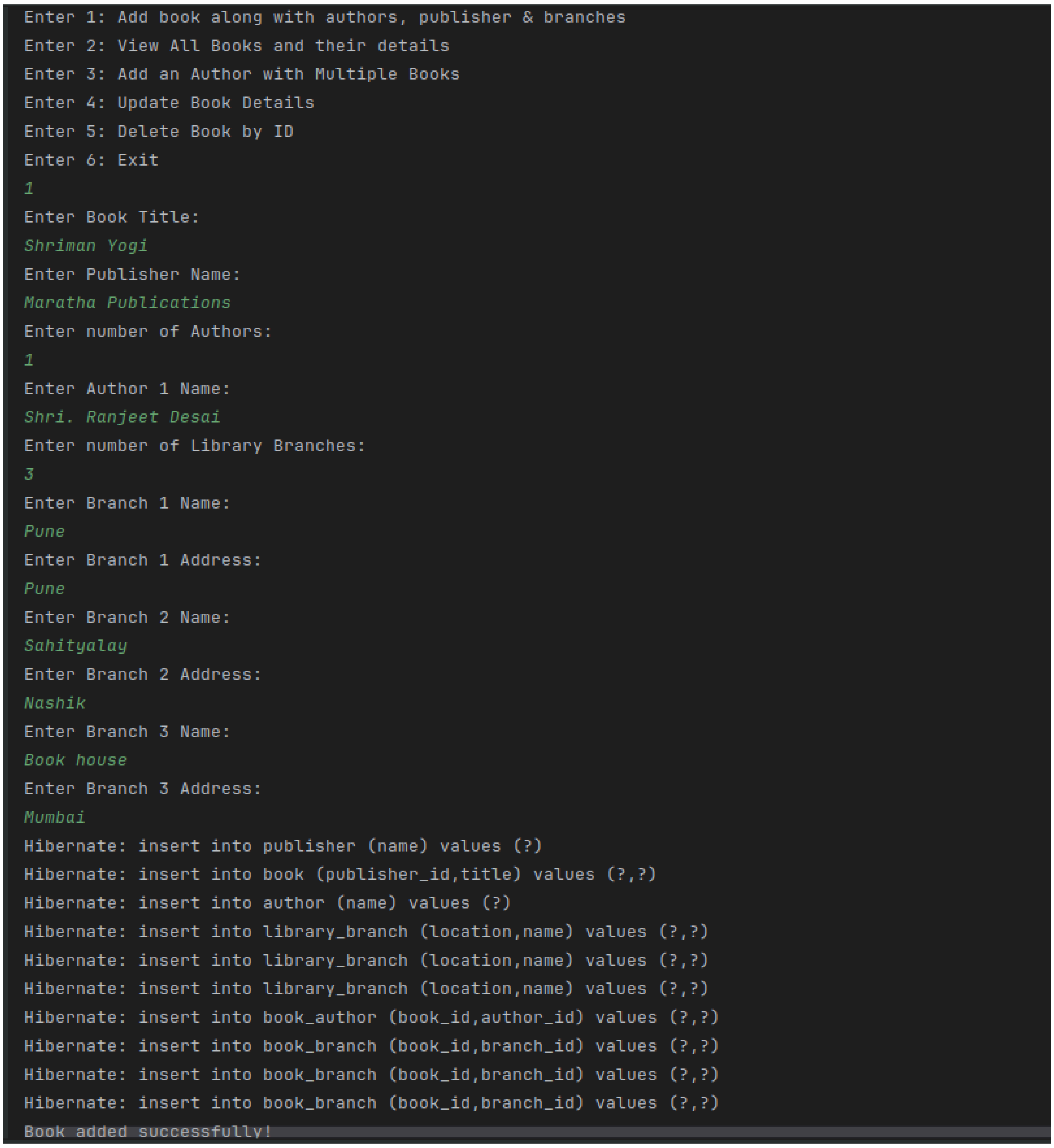
```
package org.adiitya.Scenario1;

import org.adiitya.Scenario1.beans.Author;
import org.adiitya.Scenario1.beans.Book;
import org.adiitya.Scenario1.beans.LibraryBranch;
import org.adiitya.Scenario1.beans.Publisher;
import org.adiitya.Scenario1.services.AuthorService;
import org.adiitya.Scenario1.services.BookService;
import org.adiitya.Scenario1.services.LibraryBranchService;
import org.adiitya.Scenario1.services.PublisherService;
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.ArrayList;
import java.util.List;
import java.util.Scanner;

@SpringBootApplication
public class Scenario1Application implements CommandLineRunner {

    public static void main(String[] args) {
        SpringApplication.run(Scenario1Application.class, args);
    }

    @Autowired
    private PublisherService publisherService;

    @Autowired
    private AuthorService authorService;

    @Autowired
    private BookService bookService;

    @Autowired
    private LibraryBranchService libraryBranchService;

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

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 1: Add book along with authors, publisher &
        branches");
        System.out.println("Enter 2: View All Books and their details");
        System.out.println("Enter 3: Add an Author with Multiple Books");
        System.out.println("Enter 4: Update Book Details");
        System.out.println("Enter 5: Delete Book by ID");
        System.out.println("Enter 6: Exit");

        int option = sc.nextInt();
        sc.nextLine();

        switch (option)
        {
            case 1:

                System.out.println("Enter Book Title:");
                String title = sc.nextLine();

                System.out.println("Enter Publisher Name:");
                String publisherName = sc.nextLine();
                Publisher publisher = new Publisher(publisherName);

                System.out.println("Enter number of Authors:");
                int authorCount = sc.nextInt();
                sc.nextLine();
                List<Author> authors = new ArrayList<>();
                for (int i = 1; i <= authorCount; i++) {
                    System.out.println("Enter Author "+i+" Name:");
                    String authorName = sc.nextLine();
                    Author author = new Author(authorName);
                    authors.add(author);
                }

                System.out.println("Enter number of Library Branches:");
                int branchCount = sc.nextInt();
                sc.nextLine();
                List<LibraryBranch> branches = new ArrayList<>();
                for (int i = 1; i <= branchCount; i++) {
                    System.out.println("Enter Branch "+i+" Name:");
                    String branchName = sc.nextLine();
                    System.out.println("Enter Branch "+i+" Address:");
                    String branchAddress = sc.nextLine();
                    LibraryBranch branch = new LibraryBranch(branchName, branchA
                    branches.add(branch);
                }

                Book book = new Book(title);
                book.setPublisher(publisher);
                book.setAuthors(authors);
                book.setLibraryBranches(branches);
                bookService.addBook(book);
                System.out.println("Book added successfully!");
                System.exit(0);
                break;

            case 2:

                List<Book> bookList = bookService.getAllBooks();
                if (bookList.isEmpty()) {
                    System.out.println("No books found.");
                } else {
                    for (Book b : bookList) {
                        System.out.println("------------------------------");
                        System.out.println("Title: " + b.getTitle());
                        System.out.println("Publisher: " + b.getPublisher().getName());
                        System.out.println("Authors: ");
                        for (Author a : b.getAuthors()) {
                            System.out.println(" - " + a.getName());
                        }
                        System.out.println("Branches: ");
                        for (LibraryBranch lb : b.getLibraryBranches()) {
                            System.out.println(" - " + lb.getName() + " (" + lb.getLocatior
                            + ")");
                        }
                        System.out.println("------------------------------");
                    }
                }
```

```
        System.exit(0);
        break;

    case 3:

        System.out.println("Enter Author Name:");
        String authorName = sc.nextLine();

        System.out.println("Enter number of Books for this Author:");
        int bookCount = sc.nextInt();
        sc.nextLine();

        List<Book> books = new ArrayList<>();
        for (int i = 1; i <= bookCount; i++) {
            System.out.println("Enter Details For the Book:"+i);

            System.out.println("Enter Book Title:");
            String bookTitle = sc.nextLine();

            // Publisher
            System.out.println("Enter Publisher Name of this Book:");
            String bookPublisherName = sc.nextLine();
            Publisher bookPublisher = new Publisher(bookPublisherName);

            // Branches
            System.out.println("Enter number of Library Branches for this Boo
            int bookBranchCount = sc.nextInt();
            sc.nextLine();
            List<LibraryBranch> bookBranches = new ArrayList<>();
            for (int j=1; j <= bookBranchCount; j++){
                System.out.println("Enter Branch "+j+" Name:");
                String bookBranchName = sc.nextLine();
                System.out.println("Enter Branch "+j+" Address:");
                String bookBranchAddress = sc.nextLine();
                LibraryBranch bookBranch = new LibraryBranch(bookBranchNa
                bookBranchAddress);
                bookBranches.add(bookBranch);
            }
            // Book object
            Book book1 = new Book(bookTitle);
            book1.setTitle(bookTitle);
            book1.setPublisher(bookPublisher);
            book1.setLibraryBranches(bookBranches);
            books.add(book1);
        }

        Author newAuthor = new Author(authorName);
        newAuthor.setBookList(books);

        authorService.addAuthor(newAuthor);

        System.out.println("Author with books added successfully!");
        System.exit(0);
        break;

    case 4:
        System.out.println("Enter Book ID to update:");
        int bookId = sc.nextInt();
        sc.nextLine();
        Book bookToUpdate = bookService.getBookById(bookId);
        if (bookToUpdate == null) {
            System.out.println("Book not found with ID: " + bookId);
            System.exit(0);
        }
        System.out.println("Current Title: " + bookToUpdate.getTitle());
        System.out.println("Enter new Title (or press Enter to keep current):"
        String newTitle = sc.nextLine();
        if (!newTitle.isEmpty()) {
            bookToUpdate.setTitle(newTitle);
        }
        System.out.println("Current Publisher: " + bookToUpdate.getPublish
        .getName());
        System.out.println("Enter new Publisher Name (or press Enter to kee
        current):");
        String newPublisherName = sc.nextLine();
        if (!newPublisherName.isEmpty()) {
            Publisher newPublisher = new Publisher(newPublisherName);
            bookToUpdate.setPublisher(newPublisher);
        }
        System.out.println("Current Authors: ");
        for (Author author : bookToUpdate.getAuthors()) {
            System.out.println(" - " + author.getName());
        }
        System.out.println("Enter new Authors (comma separated, or press I
        to keep current):");
        String newAuthorsInput = sc.nextLine();

        if (!newAuthorsInput.isEmpty()) {
            String[] newAuthors = newAuthorsInput.split(",");
            List<Author> authorList = new ArrayList<>();
            for (String authorNameInput : newAuthors) {
                Author author = new Author(authorNameInput.trim());
                authorList.add(author);
            }
            bookToUpdate.setAuthors(authorList);
        }
        System.out.println("Current Library Branches: ");
        for (LibraryBranch branch : bookToUpdate.getLibraryBranches()) {
            System.out.println(" - " + branch.getName() + " (" +
            branch.getLocation() + ")");
        }
        System.out.println("Enter new Library Branches (comma separated,
        or press Enter to keep current):");
        String newBranchesInput = sc.nextLine();
        if (!newBranchesInput.isEmpty()) {
            String[] newBranches = newBranchesInput.split(",");
            List<LibraryBranch> branchList = new ArrayList<>();
            for (String branchInput : newBranches) {
                String[] branchDetails = branchInput.trim().split(":");
                if (branchDetails.length == 2) {
                    LibraryBranch branch = new LibraryBranch(branchDetails[0].
                    branchDetails[1].trim());
                    branchList.add(branch);
                } else {
                    System.out.println("Invalid branch format. Use 'Name:Addres
                }
            }
            bookToUpdate.setLibraryBranches(branchList);
        }
        bookService.updateBook(bookToUpdate);
        System.out.println("Book updated successfully!");
        System.exit(0);
        break;

    case 5:
        System.out.println("Enter Book ID to delete:");
        int deleteBookId = sc.nextInt();
        sc.nextLine();

        Book bookToDelete = bookService.getBookById(deleteBookId);
        if (bookToDelete == null) {
            System.out.println("Book not found with ID: " + deleteBookId);
            System.exit(0);
        }

        bookService.deleteBook(deleteBookId);
        System.out.println("Book with ID " + deleteBookId + "
        deleted successfully!");
        System.exit(0);
        break;

    case 6:
        System.out.println("Exiting the application.");
        System.exit(0);
        break;

    default:
        System.out.println("Invalid option selected. Please try again.");
        System.exit(0);
        break;
}
```

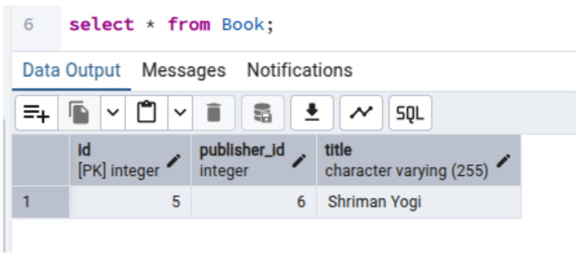
```
    }
}
```

Output:

CASE 1

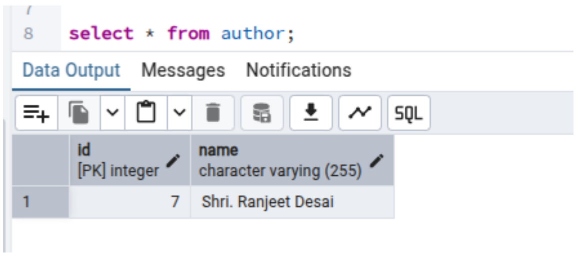


```
Enter 1: Add book along with authors, publisher & branches
Enter 2: View All Books and their details
Enter 3: Add an Author with Multiple Books
Enter 4: Update Book Details
Enter 5: Delete Book by ID
Enter 6: Exit
1
Enter Book Title:
Shriman Yogi
Enter Publisher Name:
Maratha Publications
Enter number of Authors:
1
Enter Author 1 Name:
Shri. Ranjeet Desai
Enter number of Library Branches:
3
Enter Branch 1 Name:
Pune
Enter Branch 1 Address:
Pune
Enter Branch 2 Name:
Sahityalay
Enter Branch 2 Address:
Nashik
Enter Branch 3 Name:
Book house
Enter Branch 3 Address:
Mumbai
Hibernate: insert into publisher (name) values (?)
Hibernate: insert into book (publisher_id,title) values (?,?)
Hibernate: insert into author (name) values (?)
Hibernate: insert into library_branch (location,name) values (?,?)
Hibernate: insert into library_branch (location,name) values (?,?)
Hibernate: insert into library_branch (location,name) values (?,?)
Hibernate: insert into book_author (book_id,author_id) values (?,?)
Hibernate: insert into book_branch (book_id,branch_id) values (?,?)
Hibernate: insert into book_branch (book_id,branch_id) values (?,?)
Hibernate: insert into book_branch (book_id,branch_id) values (?,?)
Book added successfully!
```



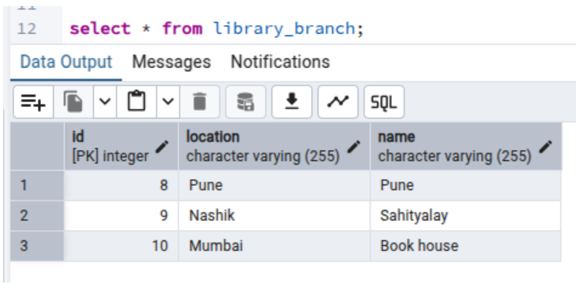
```
select * from Book;
```

	id [PK] integer	publisher_id integer	title character varying (255)
1	5	6	Shriman Yogi



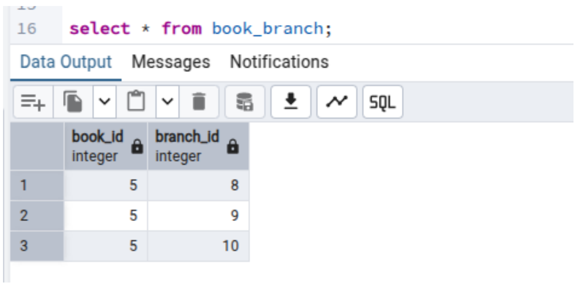
```
select * from author;
```

	id [PK] integer	name character varying (255)
1	7	Shri. Ranjeet Desai



```
select * from library_branch;
```

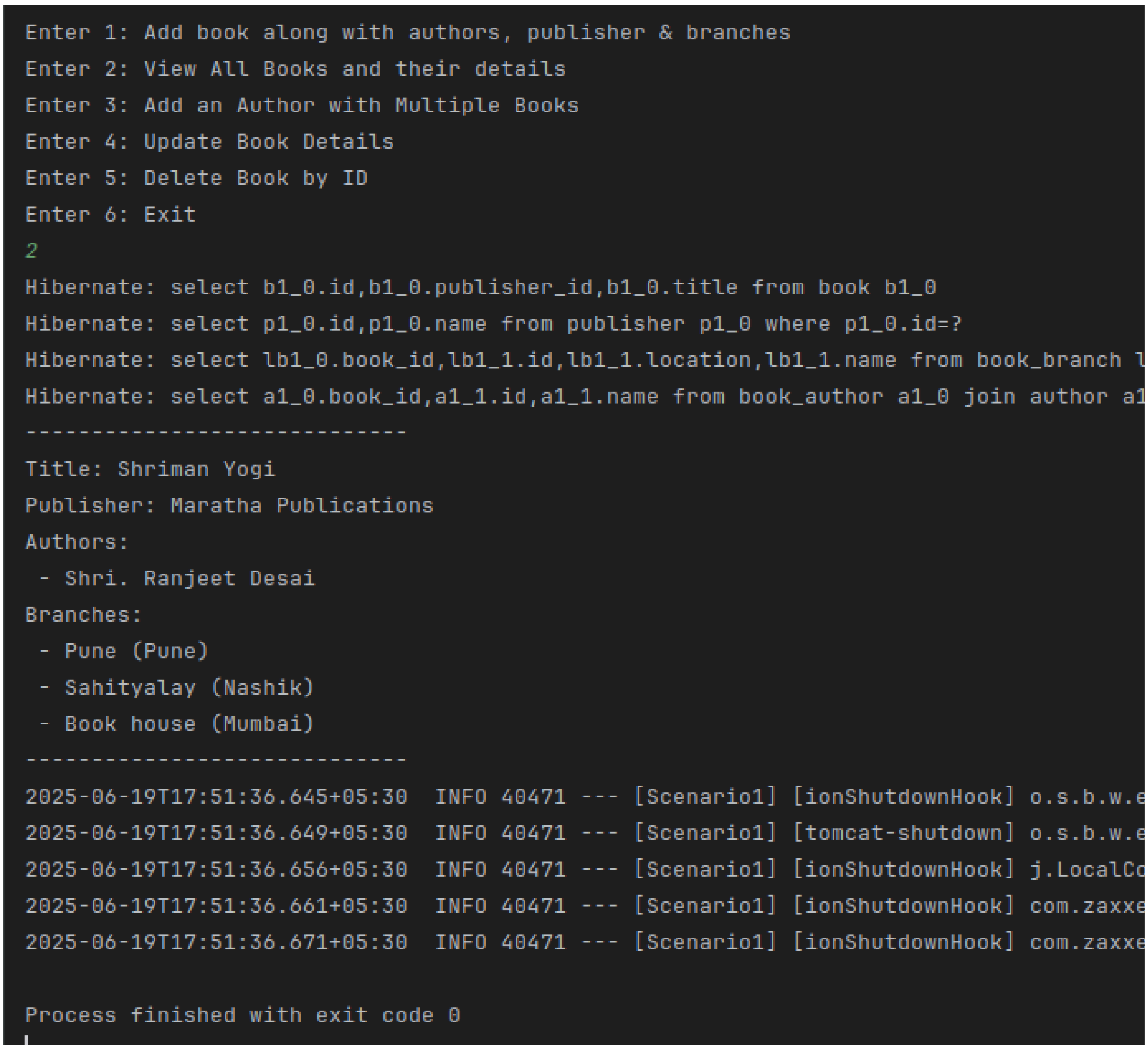
	id [PK] integer	location character varying (255)	name character varying (255)
1	8	Pune	Pune
2	9	Nashik	Sahityalay
3	10	Mumbai	Book house



```
select * from book_branch;
```

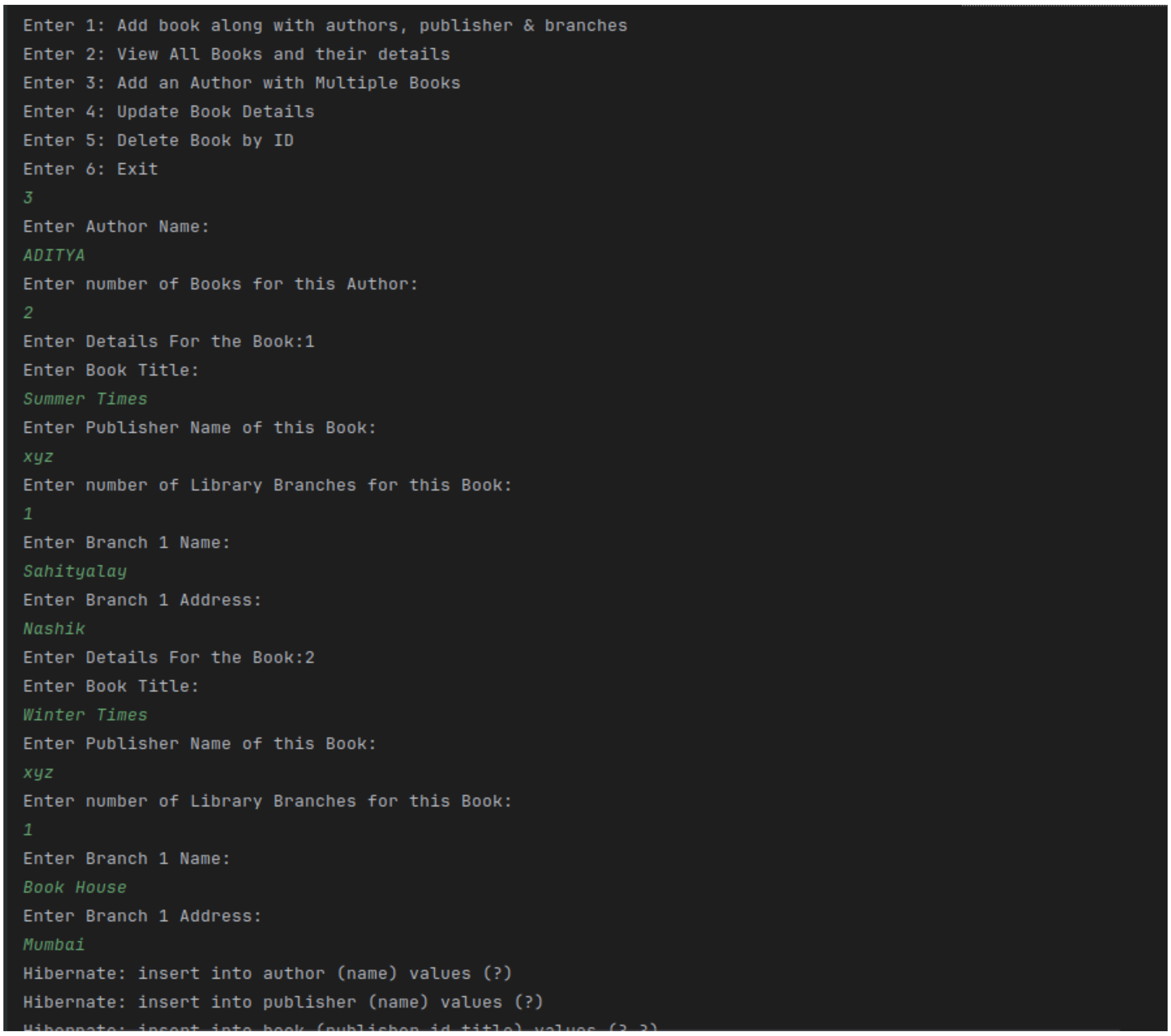
	book_id integer	branch_id integer
1	5	8
2	5	9
3	5	10

CASE 2



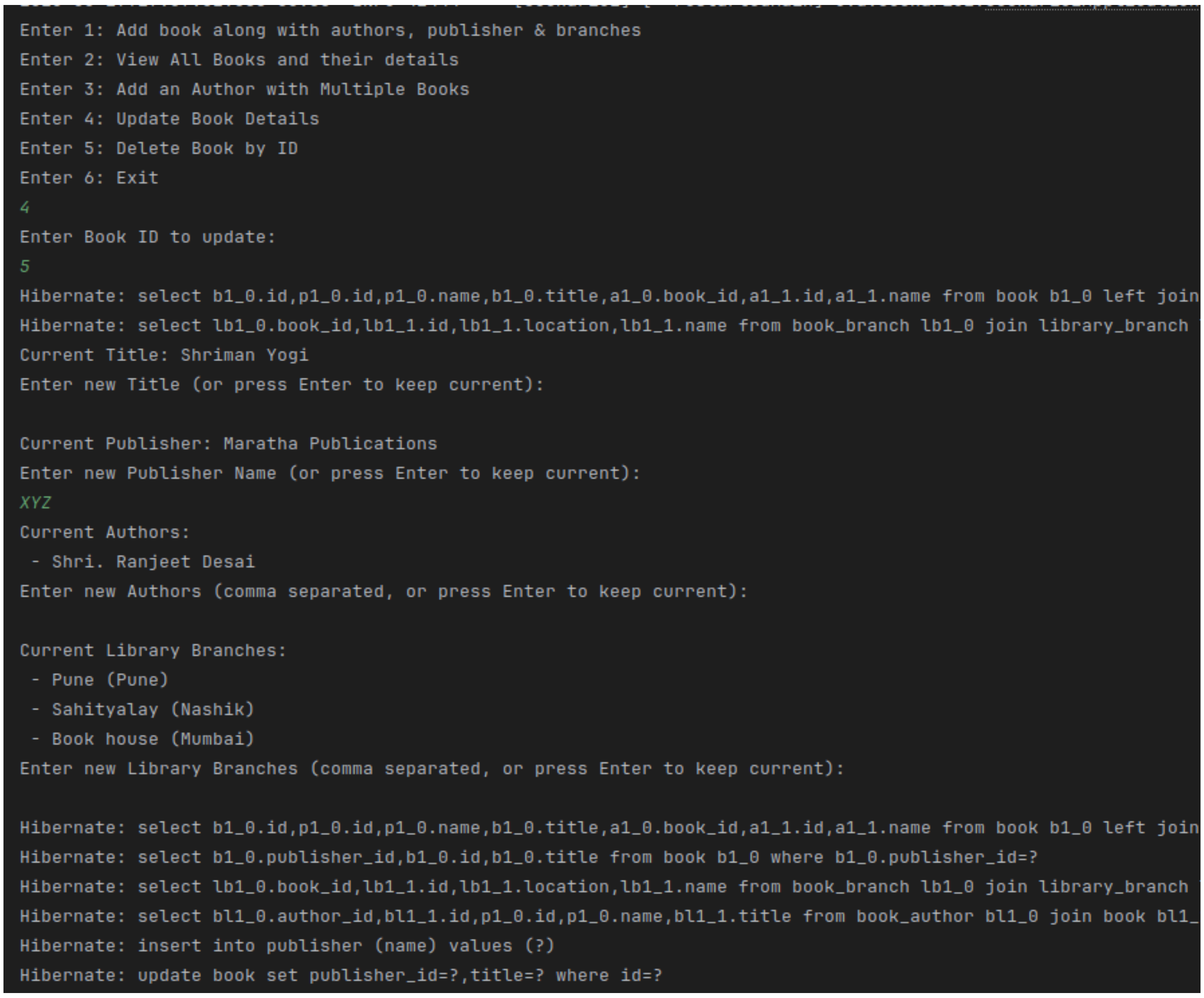
```
Enter 1: Add book along with authors, publisher & branches
Enter 2: View All Books and their details
Enter 3: Add an Author with Multiple Books
Enter 4: Update Book Details
Enter 5: Delete Book by ID
Enter 6: Exit
2
Hibernate: select b1_0.id,b1_0.publisher_id,b1_0.title from book b1_0
Hibernate: select p1_0.id,p1_0.name from publisher p1_0 where p1_0.id=?
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch l
Hibernate: select a1_0.book_id,a1_1.id,a1_1.name from book_author a1_0 join author a1
-----------------------------
Title: Shriman Yogi
Publisher: Maratha Publications
Authors:
 - Shri. Ranjeet Desai
Branches:
 - Pune (Pune)
 - Sahityalay (Nashik)
 - Book house (Mumbai)
-----------------------------
2025-06-19T17:51:36.645+05:30  INFO 40471 --- [Scenario1] [ionShutdownHook] o.s.b.w.e
2025-06-19T17:51:36.649+05:30  INFO 40471 --- [Scenario1] [tomcat-shutdown] o.s.b.w.e
2025-06-19T17:51:36.656+05:30  INFO 40471 --- [Scenario1] [ionShutdownHook] j.LocalCo
2025-06-19T17:51:36.661+05:30  INFO 40471 --- [Scenario1] [ionShutdownHook] com.zaxxe
2025-06-19T17:51:36.671+05:30  INFO 40471 --- [Scenario1] [ionShutdownHook] com.zaxxe

Process finished with exit code 0
```

CASE 3



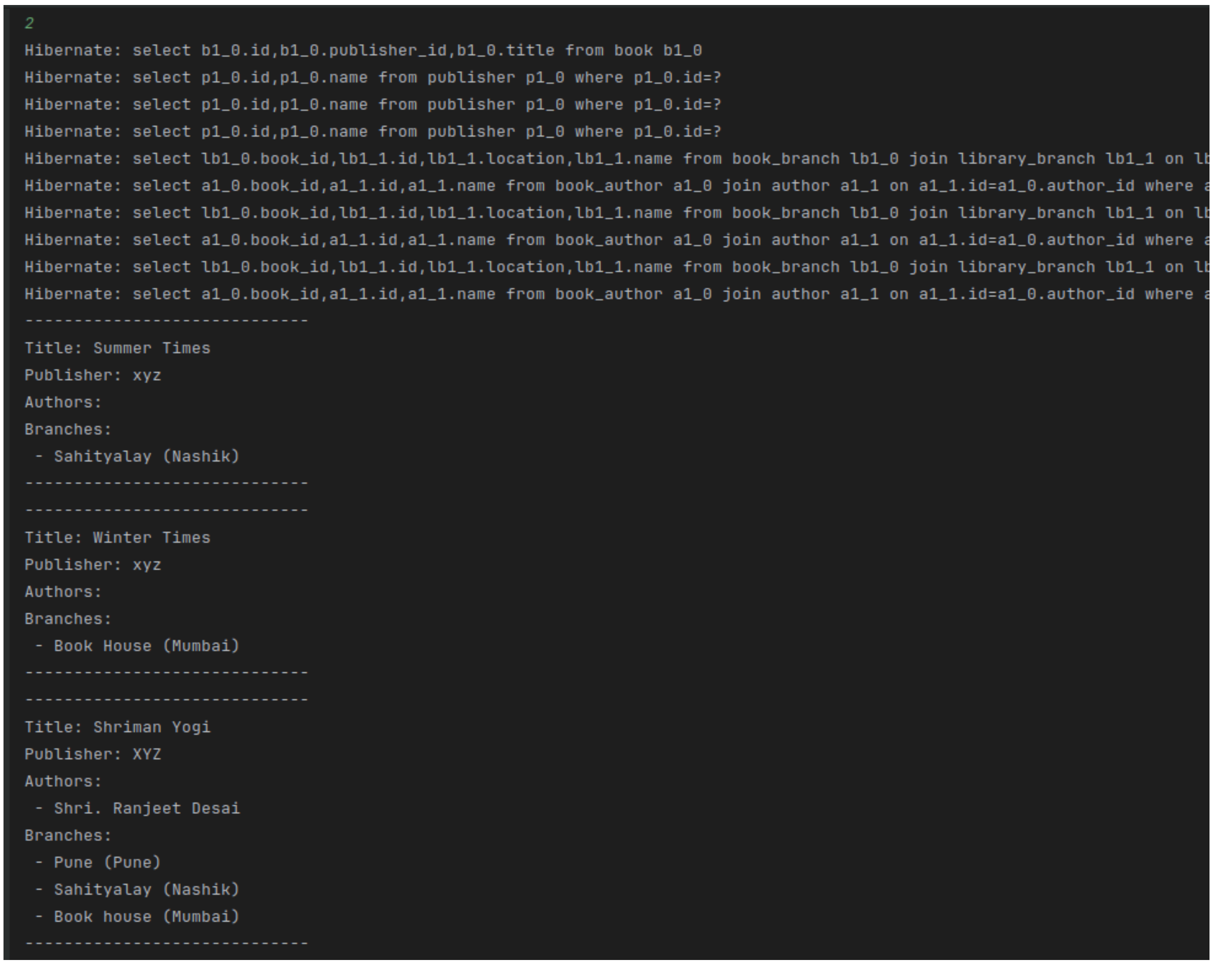
```
Enter 1: Add book along with authors, publisher & branches
Enter 2: View All Books and their details
Enter 3: Add an Author with Multiple Books
Enter 4: Update Book Details
Enter 5: Delete Book by ID
Enter 6: Exit
3
Enter Author Name:
ADITYA
Enter number of Books for this Author:
2
Enter Details For the Book:1
Enter Book Title:
Summer Times
Enter Publisher Name of this Book:
xyz
Enter number of Library Branches for this Book:
1
Enter Branch 1 Name:
Sahityalay
Enter Branch 1 Address:
Nashik
Enter Details For the Book:2
Enter Book Title:
Winter Times
Enter Publisher Name of this Book:
xyz
Enter number of Library Branches for this Book:
1
Enter Branch 1 Name:
Book House
Enter Branch 1 Address:
Mumbai
Hibernate: insert into author (name) values (?)
Hibernate: insert into publisher (name) values (?)
Hibernate: insert into book (publisher_id,title) values (?,?)
```

CASE 4



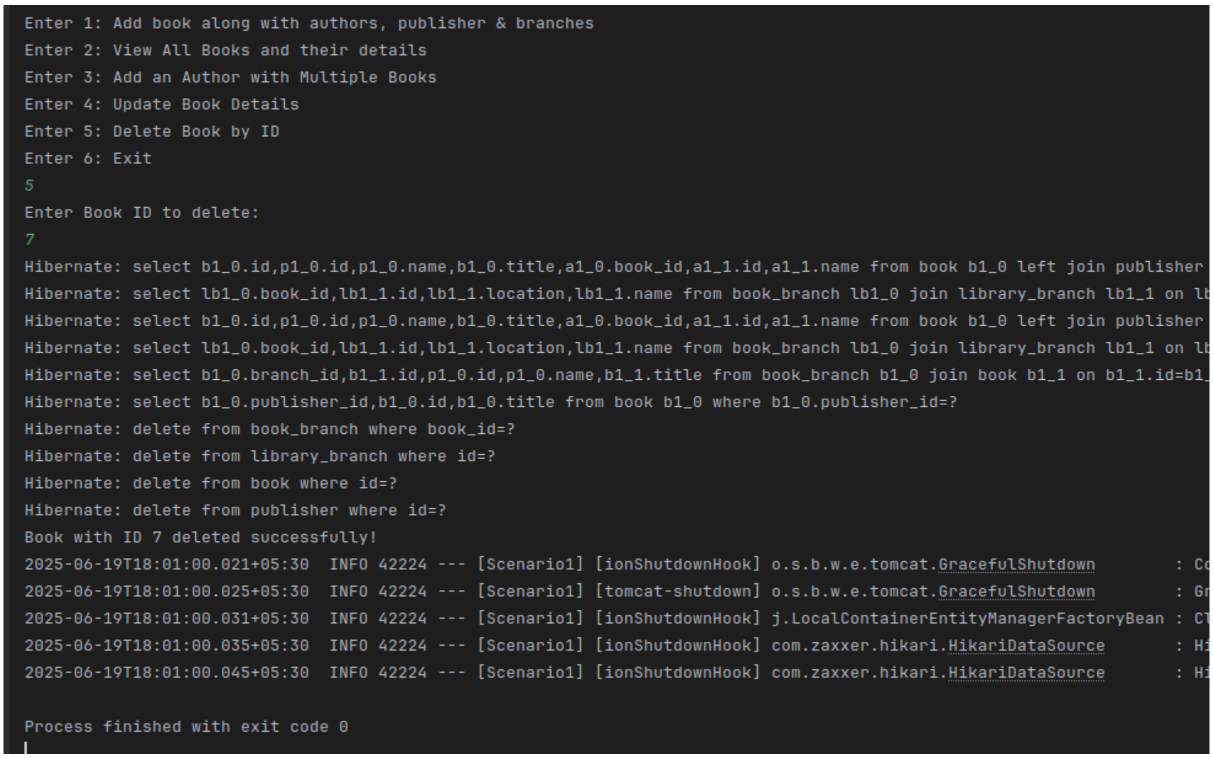
```
Enter 1: Add book along with authors, publisher & branches
Enter 2: View All Books and their details
Enter 3: Add an Author with Multiple Books
Enter 4: Update Book Details
Enter 5: Delete Book by ID
Enter 6: Exit
4
Enter Book ID to update:
5
Hibernate: select b1_0.id,p1_0.id,p1_0.name,b1_0.title,a1_0.book_id,a1_1.id,a1_1.name from book b1_0 left join
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch
Current Title: Shriman Yogi
Enter new Title (or press Enter to keep current):

Current Publisher: Maratha Publications
Enter new Publisher Name (or press Enter to keep current):
XYZ
Current Authors:
 - Shri. Ranjeet Desai
Enter new Authors (comma separated, or press Enter to keep current):

Current Library Branches:
 - Pune (Pune)
 - Sahityalay (Nashik)
 - Book house (Mumbai)
Enter new Library Branches (comma separated, or press Enter to keep current):

Hibernate: select b1_0.id,p1_0.id,p1_0.name,b1_0.title,a1_0.book_id,a1_1.id,a1_1.name from book b1_0 left join
Hibernate: select b1_0.publisher_id,b1_0.id,b1_0.title from book b1_0 where b1_0.publisher_id=?
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch
Hibernate: select bl1_0.author_id,bl1_1.id,p1_0.id,p1_0.name,bl1_1.title from book_author bl1_0 join book bl1_
Hibernate: insert into publisher (name) values (?)
Hibernate: update book set publisher_id=?,title=? where id=?
```

view books



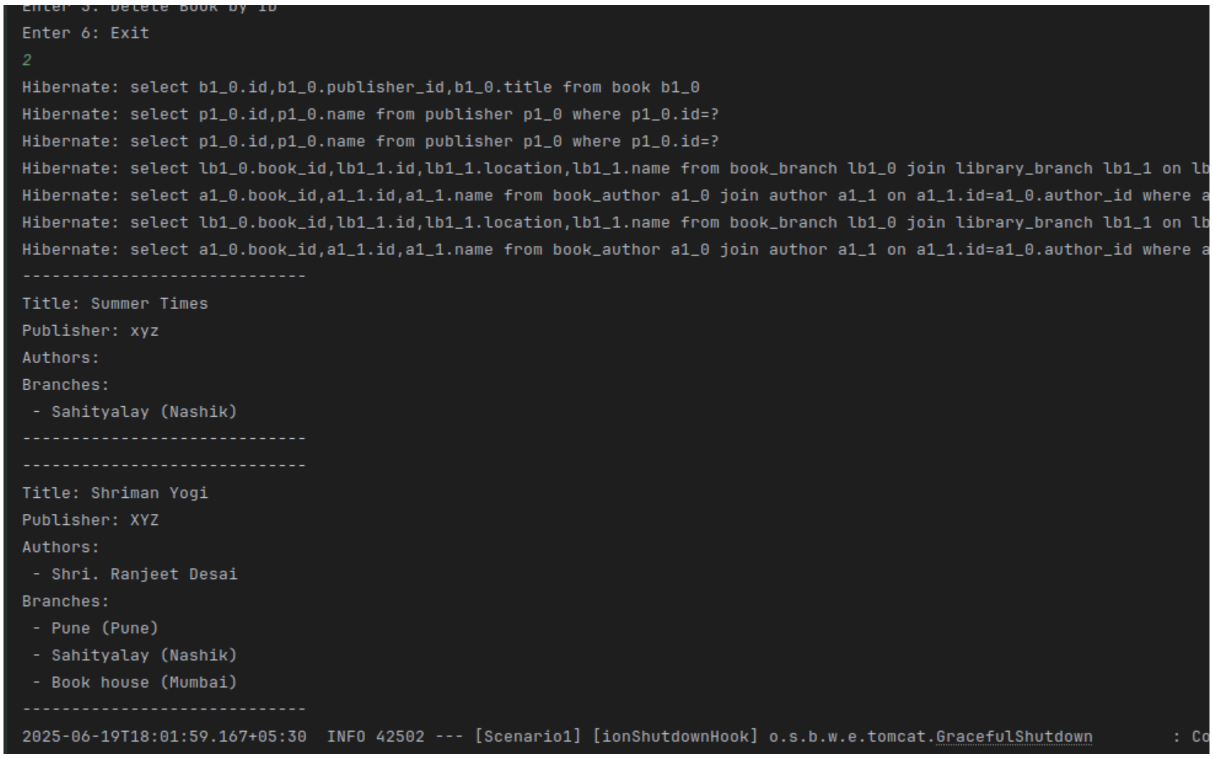
```
2
Hibernate: select b1_0.id,b1_0.publisher_id,b1_0.title from book b1_0
Hibernate: select p1_0.id,p1_0.name from publisher p1_0 where p1_0.id=?
Hibernate: select p1_0.id,p1_0.name from publisher p1_0 where p1_0.id=?
Hibernate: select p1_0.id,p1_0.name from publisher p1_0 where p1_0.id=?
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch lb1_1 on lb
Hibernate: select a1_0.book_id,a1_1.id,a1_1.name from book_author a1_0 join author a1_1 on a1_1.id=a1_0.author_id where a
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch lb1_1 on lb
Hibernate: select a1_0.book_id,a1_1.id,a1_1.name from book_author a1_0 join author a1_1 on a1_1.id=a1_0.author_id where a
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch lb1_1 on lb
Hibernate: select a1_0.book_id,a1_1.id,a1_1.name from book_author a1_0 join author a1_1 on a1_1.id=a1_0.author_id where a
-----------------------------
Title: Summer Times
Publisher: xyz
Authors:
Branches:
 - Sahityalay (Nashik)
-----------------------------
-----------------------------
Title: Winter Times
Publisher: xyz
Authors:
Branches:
 - Book House (Mumbai)
-----------------------------
-----------------------------
Title: Shriman Yogi
Publisher: XYZ
Authors:
 - Shri. Ranjeet Desai
Branches:
 - Pune (Pune)
 - Sahityalay (Nashik)
 - Book house (Mumbai)
-----------------------------
```

CASE 5



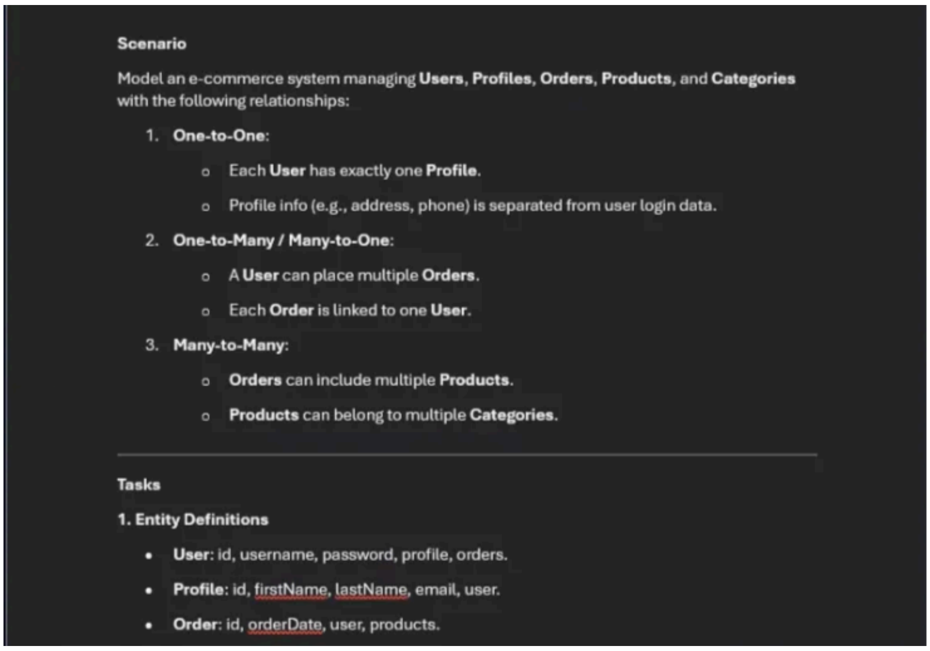
```
Enter 1: Add book along with authors, publisher & branches
Enter 2: View All Books and their details
Enter 3: Add an Author with Multiple Books
Enter 4: Update Book Details
Enter 5: Delete Book by ID
Enter 6: Exit
5
Enter Book ID to delete:
7
Hibernate: select b1_0.id,p1_0.id,p1_0.name,b1_0.title,a1_0.book_id,a1_1.id,a1_1.name from book b1_0 left join publisher
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch lb1_1 on lb
Hibernate: select b1_0.id,p1_0.id,p1_0.name,b1_0.title,a1_0.book_id,a1_1.id,a1_1.name from book b1_0 left join publisher
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch lb1_1 on lb
Hibernate: select b1_0.branch_id,b1_1.id,p1_0.id,p1_0.name,b1_1.title from book_branch b1_0 join book b1_1 on b1_1.id=b1_
Hibernate: select b1_0.publisher_id,b1_0.id,b1_0.title from book b1_0 where b1_0.publisher_id=?
Hibernate: delete from book_branch where book_id=?
Hibernate: delete from library_branch where id=?
Hibernate: delete from book where id=?
Hibernate: delete from publisher where id=?
Book with ID 7 deleted successfully!
2025-06-19T18:01:00.021+05:30  INFO 42224 --- [Scenario1] [ionShutdownHook] o.s.b.w.e.tomcat.GracefulShutdown        : Co
2025-06-19T18:01:00.025+05:30  INFO 42224 --- [Scenario1] [tomcat-shutdown] o.s.b.w.e.tomcat.GracefulShutdown        : Gr
2025-06-19T18:01:00.031+05:30  INFO 42224 --- [Scenario1] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Cl
2025-06-19T18:01:00.035+05:30  INFO 42224 --- [Scenario1] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : Hi
2025-06-19T18:01:00.045+05:30  INFO 42224 --- [Scenario1] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : Hi

Process finished with exit code 0
```

view books



```
Enter 5: Delete Book by ID
Enter 6: Exit
2
Hibernate: select b1_0.id,b1_0.publisher_id,b1_0.title from book b1_0
Hibernate: select p1_0.id,p1_0.name from publisher p1_0 where p1_0.id=?
Hibernate: select p1_0.id,p1_0.name from publisher p1_0 where p1_0.id=?
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch lb1_1 on lb
Hibernate: select a1_0.book_id,a1_1.id,a1_1.name from book_author a1_0 join author a1_1 on a1_1.id=a1_0.author_id where a
Hibernate: select lb1_0.book_id,lb1_1.id,lb1_1.location,lb1_1.name from book_branch lb1_0 join library_branch lb1_1 on lb
Hibernate: select a1_0.book_id,a1_1.id,a1_1.name from book_author a1_0 join author a1_1 on a1_1.id=a1_0.author_id where a
-----------------------------
Title: Summer Times
Publisher: xyz
Authors:
Branches:
 - Sahityalay (Nashik)
-----------------------------
-----------------------------
Title: Shriman Yogi
Publisher: XYZ
Authors:
 - Shri. Ranjeet Desai
Branches:
 - Pune (Pune)
 - Sahityalay (Nashik)
 - Book house (Mumbai)
-----------------------------
2025-06-19T18:01:59.167+05:30  INFO 42502 --- [Scenario1] [ionShutdownHook] o.s.b.w.e.tomcat.GracefulShutdown        : Co
```

Scenario2:



Scenario

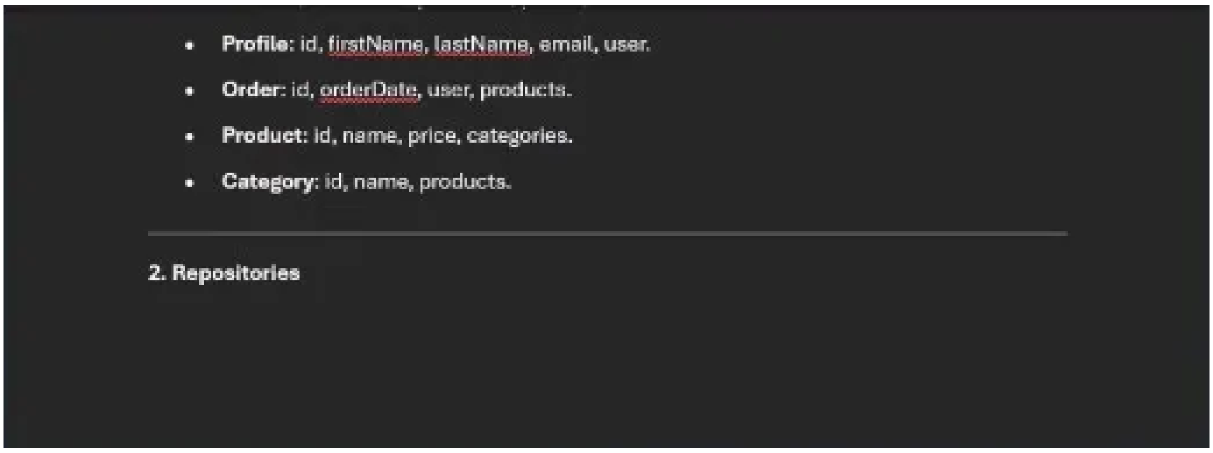
Model an e-commerce system managing **Users**, **Profiles**, **Orders**, **Products**, and **Categories** with the following relationships:

1. **One-to-One**:
   - Each **User** has exactly one **Profile**.
   - Profile info (e.g., address, phone) is separated from user login data.
2. **One-to-Many / Many-to-One**:
   - A **User** can place multiple **Orders**.
   - Each **Order** is linked to one **User**.
3. **Many-to-Many**:
   - **Orders** can include multiple **Products**.
   - **Products** can belong to multiple **Categories**.

Tasks

1. Entity Definitions

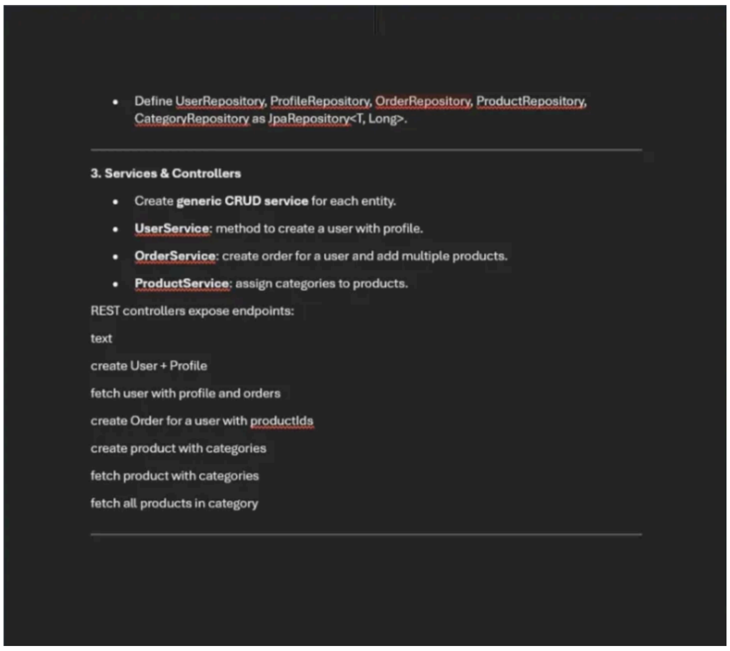
- **User**: id, username, password, profile, orders.
- **Profile**: id, firstName, lastName, email, user.
- **Order**: id, orderDate, user, products.



- **Profile**: id, firstName, lastName, email, user.
- **Order**: id, orderDate, user, products.
- **Product**: id, name, price, categories.
- **Category**: id, name, products.

---

2. Repositories



- Define UserRepository, ProfileRepository, OrderRepository, ProductRepository, CategoryRepository as JpaRepository<T, Long>.

---

3. Services & Controllers

- Create **generic CRUD service** for each entity.
- **UserService**: method to create a user with profile.
- **OrderService**: create order for a user and add multiple products.
- **ProductService**: assign categories to products.

REST controllers expose endpoints:

text

create User + Profile

fetch user with profile and orders

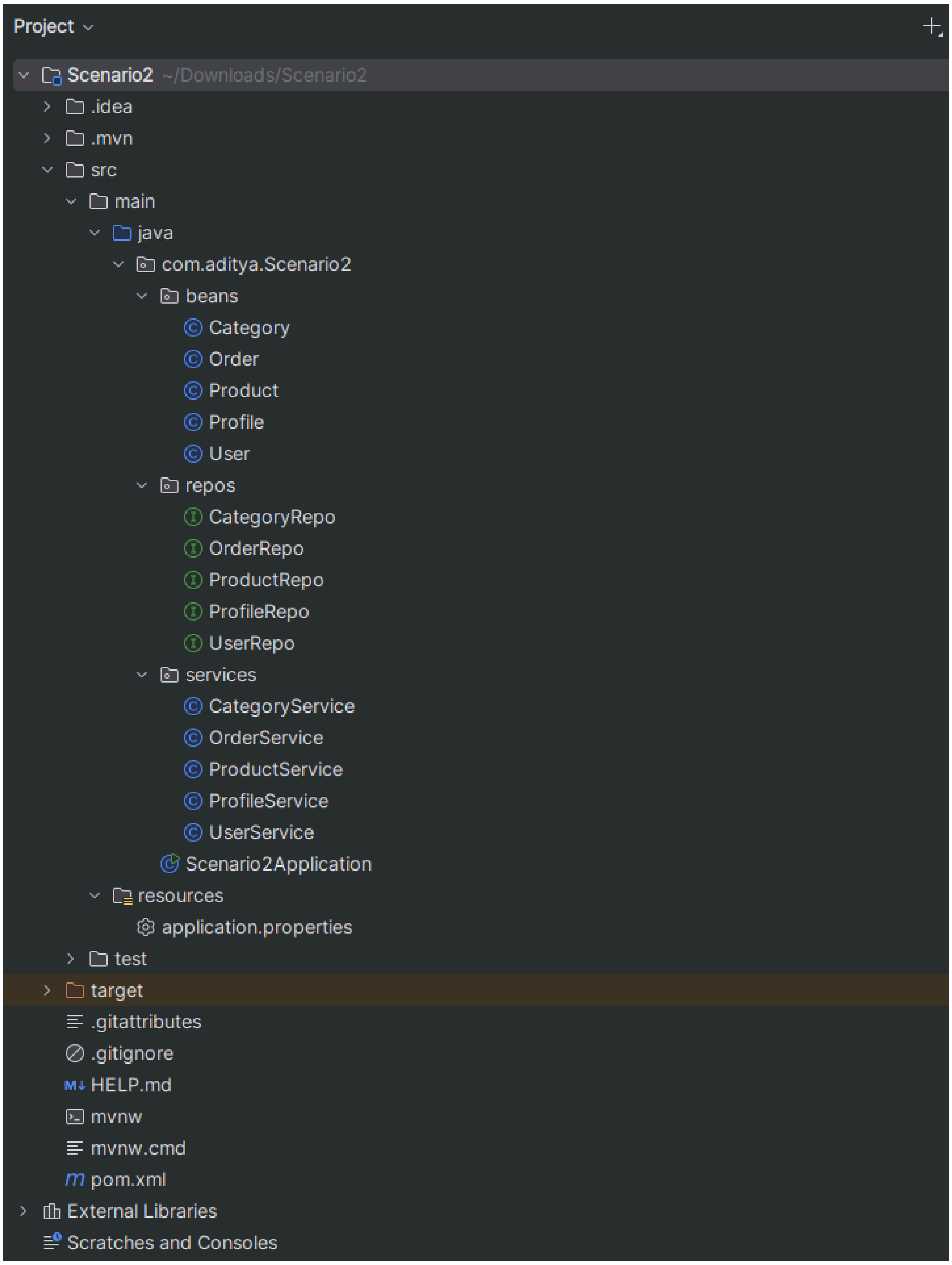
create Order for a user with productIds

create product with categories

fetch product with categories

fetch all products in category

Folder Structure:



```
Project
Scenario2 ~/Downloads/Scenario2
  .idea
  .mvn
  src
    main
      java
        com.aditya.Scenario2
          beans
            Category
            Order
            Product
            Profile
            User
          repos
            CategoryRepo
            OrderRepo
            ProductRepo
            ProfileRepo
            UserRepo
          services
            CategoryService
            OrderService
            ProductService
            ProfileService
            UserService
          Scenario2Application
      resources
        application.properties
    test
  target
  .gitattributes
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

POM.XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
```

```xml
/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.ap
    .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>3.5.0</version>
        <relativePath/> <!-- lookup parent from repository →
    </parent>
    <groupId>com.aditya</groupId>
    <artifactId>Scenario2</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Scenario2</name>
    <description>Demo project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>21</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Application.properties:

```
spring.application.name=Scenario2
# ===============================
# DATABASE CONFIGURATION
# ================================
spring.datasource.url=jdbc:postgresql://localhost:5432/assignment3
spring.datasource.username=postgres
spring.datasource.password=root

# Load database Driver
#spring.datasource.driver-class-name=org.postgresql.Driver

# ================================
# JPA / HIBERNATE CONFIGURATION
# ================================
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=false

# Naming strategy (CamelCase → snake_case table/column names)
#spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.nar
.PhysicalNamingStrategyStandardImpl

# ================================
# SERVER SETTINGS
# ================================
server.port=8080

spring.data.jpa.repositories.bootstrap-mode=default
spring.data.defer-datasource-initalization=tru
```

User.java

```
package com.aditya.Scenario2.beans;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;

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

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

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int userId;
    @NotBlank(message = "Name cannot be empty")
    private String userName;
    @NotBlank(message = "password cannot be empty")
    private String password;

    // one-to-one relationship... User ⟵⟶> Profile
    @OneToOne(mappedBy = "user" ,cascade = CascadeType.ALL,
    fetch = FetchType.EAGER)
    private Profile profile;

    // one-to-many relationship... User ⟵⟶>> Order
    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL,
    fetch = FetchType.EAGER)
    private List<Order> orders = new ArrayList<>();

    // constructors
    public User(){}

    public User(String userName, String password){
        this.userName = userName;
        this.password = password;
    }

    // getters and setters
    public int getId() {
        return userId;
    }

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

    public List<Order> getOrders() {
        return orders;
    }

    public void setOrders(List<Order> orders) {
        this.orders = orders;
    }

    public Profile getProfile() {
        return profile;
    }

    public void setProfile(Profile profile) {
        this.profile = profile;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }


    // ToString method
    @Override
    public String toString() {
        return "Users{" +
                "userName='" + userName + '\'' +
                ", password='" + password + '\'' +
                ", id=" + userId +
                '}';
    }
}
```

Profile.java

```
package com.aditya.Scenario2.beans;

import jakarta.persistence.*;

@Entity
@Table(name = "profiles")
public class Profile {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int profileId;
    private String firstName;
    private String lastName;
    private String email;
    private String address;
    private long phoneNumber;

    @OneToOne(fetch = FetchType.EAGER)
    // one-to-one relationship... Profile <——> User
    @JoinColumn(name = "user_id", nullable = false, unique = true)
    private User user;

    // Constructors
    public Profile(){}
    public Profile(String firstName, String lastName, String email,
    String address, long phoneNumber){
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.address = address;
        this.phoneNumber = phoneNumber;
    }

    // getter and setters
    public int getProfileId() {
        return profileId;
    }

    public void setProfileId(int profileId) {
        this.profileId = profileId;
    }

    public User getUser() {
        return user;
    }

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

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public long getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(long phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
```

```
public String getEmail() {
    return email;
}

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

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}


// ToString
@Override
public String toString() {
    return "Profile{" +
            "firstName='" + firstName + '\'' +
            ", profileId=" + profileId +
            ", lastName='" + lastName + '\'' +
            ", email='" + email + '\'' +
            ", address='" + address + '\'' +
            ", phoneNumber=" + phoneNumber +
            '}';
    }
}
```

Product.java

```
package com.aditya.Scenario2.beans;

import jakarta.persistence.*;

import java.util.List;

@Entity
@Table(name = "products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private double price;

    // Bidirectional Many-to-Many with Order
    @ManyToMany(mappedBy = "products")
    private List<Order> orders;

    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinTable(
            name = "product_category",
            joinColumns = @JoinColumn(name = "product_id"),
            inverseJoinColumns = @JoinColumn(name = "category_id")
    )
    private List<Category> categories;

    // constructors
    public Product() {
    }
    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    // Getters and Setters


    public int getId() {
        return id;
    }

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

    public List<Order> getOrders() {
        return orders;
    }

    public void setOrders(List<Order> orders) {
        this.orders = orders;
    }

    public List<Category> getCategories() {
        return categories;
    }

    public void setCategories(List<Category> categories) {
        this.categories = categories;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
    public String getName() {
        return name;
    }

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

    // ToString
    @Override
    public String toString() {
        return "Product{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", price=" + price +
                '}';
    }
}
```

Order.java

```
package com.aditya.Scenario2.beans;

import jakarta.persistence.*;

import java.util.Date;
import java.util.List;

@Entity
@Table(name = "orders")
public class Order {

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

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinTable(
        name = "order_product",
        joinColumns = @JoinColumn(name = "order_id"),
        inverseJoinColumns = @JoinColumn(name = "product_id")
    )
    private List<Product> products;

    // constructors
    public Order() {
    }

    public Order(Date orderDate) {
        this.orderDate = orderDate;
    }

   // Getters and Setters


    public int getId() {
        return id;
    }

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

    public User getUser() {
        return user;
    }

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

    public List<Product> getProducts() {
        return products;
    }

    public void setProducts(List<Product> products) {
        this.products = products;
    }

    public Date getOrderDate() {
        return orderDate;
    }

    public void setOrderDate(Date orderDate) {
        this.orderDate = orderDate;
    }

    public void addProduct(Product product) {
        this.products.add(product);
        product.getOrders().add(this);
    }

    public void removeProduct(Product product) {
        products.remove(product);
        product.getOrders().remove(this);
    }

    // ToString
    @Override
    public String toString() {
        return "Order{" +
                "id=" + id +
                ", orderDate=" + orderDate +
                '}';
    }
}
```

Category.java

```
package com.aditya.Scenario2.beans;

import jakarta.persistence.*;

import java.util.List;

@Entity
@Table(name = "categories")
public class Category {

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

    private String name;

    @ManyToMany(mappedBy = "categories", cascade = CascadeType.ALL,
    fetch = FetchType.EAGER)
    private List<Product> products;

    // constructors
    public Category() {
    }
    public Category(String name) {
        this.name = name;
    }

    // Getters and Setters

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
    public List<Product> getProducts() {
        return products;
    }

    public void setProducts(List<Product> products) {
        this.products = products;
    }

    public String getName() {
        return name;
    }

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

    public void addProduct(Product product) {
        products.add(product);
        product.getCategories().add(this);
    }

    public void removeProduct(Product product) {
        products.remove(product);
        product.getCategories().remove(this);
    }

    @Override
    public String toString() {
        return "Category{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
}
```

Repos:

```
//***********************************************************************
package com.aditya.Scenario2.repos;

import com.aditya.Scenario2.beans.Category;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CategoryRepo extends JpaRepository<Category,Integer> {
}
//***********************************************************************

package com.aditya.Scenario2.repos;

import com.aditya.Scenario2.beans.Order;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface OrderRepo extends JpaRepository<Order,Integer> {
}

//***********************************************************************

package com.aditya.Scenario2.repos;

import com.aditya.Scenario2.beans.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ProductRepo extends JpaRepository<Product,Integer> {
}

//***********************************************************************

package com.aditya.Scenario2.repos;
```

```
import com.aditya.Scenario2.beans.Profile;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ProfileRepo extends JpaRepository<Profile,Integer> {
}

//*********************************************************************

package com.aditya.Scenario2.repos;

import com.aditya.Scenario2.beans.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepo extends JpaRepository<User,Integer> {
}

//*********************************************************************
```

UserService.java

```
package com.aditya.Scenario2.services;

import com.aditya.Scenario2.beans.User;
import com.aditya.Scenario2.repos.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserService {

    @Autowired
    private UserRepo userRepo;

    public void addUser(User user) {
        userRepo.save(user);
    }

    public User getUserById(int id) {
        return userRepo.findById(id).orElse(null);
    }

    public List<User> getAllUsers() {
        return userRepo.findAll();
    }

    public void deleteUser(int id) {
        userRepo.deleteById(id);
    }

    public User updateUser(User user) {
        return userRepo.save(user);
    }
}
```

ProfileService.java

```
package com.aditya.Scenario2.services;

import com.aditya.Scenario2.beans.Profile;
import com.aditya.Scenario2.repos.ProfileRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class ProfileService {

    @Autowired
    private ProfileRepo profileRepo;
    public void addProfile(Profile profile) {
        profileRepo.save(profile);
    }

    public Profile getProfileById(int id) {
        return profileRepo.findById(id).orElse(null);
    }

    public void deleteProfile(int id) {
        profileRepo.deleteById(id);
    }

    public Profile updateProfile(Profile profile) {
        return profileRepo.save(profile);
    }

}
```

ProductService.java

```
package com.aditya.Scenario2.services;

import com.aditya.Scenario2.beans.Product;
import com.aditya.Scenario2.repos.ProductRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ProductService {

    @Autowired
    private ProductRepo productRepo;

    // Add methods for CRUD operations on Product
    public void addProduct(Product product) {
        productRepo.save(product);
    }
    public Product getProductById(int id) {
        return productRepo.findById(id).orElse(null);
    }
    public void deleteProduct(int id) {
        productRepo.deleteById(id);
    }
    public Product updateProduct(Product product) {
        return productRepo.save(product);
    }
    public List<Product> getAllProducts() {
        return productRepo.findAll();
    }

    public Product returnProduct(Product product){
        return product;
    }

}
```

OrderService.java

```
package com.aditya.Scenario2.services;

import com.aditya.Scenario2.beans.Order;
import com.aditya.Scenario2.beans.User;
import com.aditya.Scenario2.repos.OrderRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class OrderService {
    @Autowired
    private OrderRepo orderRepo;

    // Add methods for CRUD operations on Order
    public void addOrder(Order order) {
        orderRepo.save(order);
    }
    public Order getOrderById(int id) {
        return orderRepo.findById(id).orElse(null);
    }
    public void deleteOrder(int id) {
        orderRepo.deleteById(id);
    }
    public Order updateOrder(Order order) {
        return orderRepo.save(order);
    }
    public List<Order> getAllOrders() {
        return orderRepo.findAll();
    }

}
```

CategoryService.java

```
package com.aditya.Scenario2.services;

import com.aditya.Scenario2.beans.Category;
import com.aditya.Scenario2.beans.Product;
import com.aditya.Scenario2.repos.CategoryRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Collections;
import java.util.List;

@Service
public class CategoryService {
    private final CategoryRepo categoryRepo;

    @Autowired
    public CategoryService(CategoryRepo categoryRepo) {
        this.categoryRepo = categoryRepo;
    }

    public List<Product> getProductsInCategory(int categoryId) {
        return categoryRepo.findById(categoryId)
                .map(Category::getProducts)
                .orElse(Collections.emptyList());
    }

}
```

Scenario2Application.java

```
package com.aditya.Scenario2;

import com.aditya.Scenario2.beans.*;
import com.aditya.Scenario2.services.*;
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.ArrayList;
import java.util.List;
import java.util.Scanner;

@SpringBootApplication
public class Scenario2Application implements CommandLineRunner {

    public static void main(String[] args) {
        SpringApplication.run(Scenario2Application.class, args);
    }
```

```
@Autowired
private UserService userService;

@Autowired
private ProfileService profileService;

@Autowired
private ProductService productService;

@Autowired
private OrderService orderService;

@Autowired
private CategoryService categoryService;

@Override
public void run(String... args) throws Exception {
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter 1: Add User along with Profile");
    System.out.println("Enter 2: View User with Profile and Orders");
    System.out.println("Enter 3: Create Order for User");
    System.out.println("Enter 4: Create Product with Categories");
    System.out.println("Enter 5: View Product with Categories");
    System.out.println("Enter 6: View All Products in a Category");
    System.out.println("Enter 7: View All Orders");

    System.out.println("Enter 0: Exit");

    int option = sc.nextInt();
    sc.nextLine();

    switch (option) {
        case 0:
            System.out.println("Exiting the application.");
            System.exit(0);
            break;
```

```
case 1:
    System.out.println("Enter Username:");
    String username = sc.nextLine();
    System.out.println("Enter Password:");
    String password = sc.nextLine();
    System.out.println("Enter First Name:");
    String firstName = sc.nextLine();
    System.out.println("Enter Last Name:");
    String lastName = sc.nextLine();
    System.out.println("Enter Email:");
    String email = sc.nextLine();
    System.out.println("Enter Address:");
    String address = sc.nextLine();
    System.out.println("Enter Phone Number:");
    long phoneNumber = sc.nextLong();
    sc.nextLine(); // consume newline

    User user = new User(username, password);
    Profile profile = new Profile(firstName, lastName, email, address,
    phoneNumber);
    user.setProfile(profile);
    profile.setUser(user);

    userService.addUser(user);
    System.out.println("User and Profile created successfully!");
    System.exit(0);
    break;

case 2:
    System.out.println("Enter User ID to view details:");
    int userId = sc.nextInt();
    sc.nextLine(); // consume newline
    User foundUser = userService.getUserById(userId);
    if (foundUser != null) {
        System.out.println("**********************************************
        System.out.println("User Details:");
        System.out.println("Username: " + foundUser.getUserName());
```

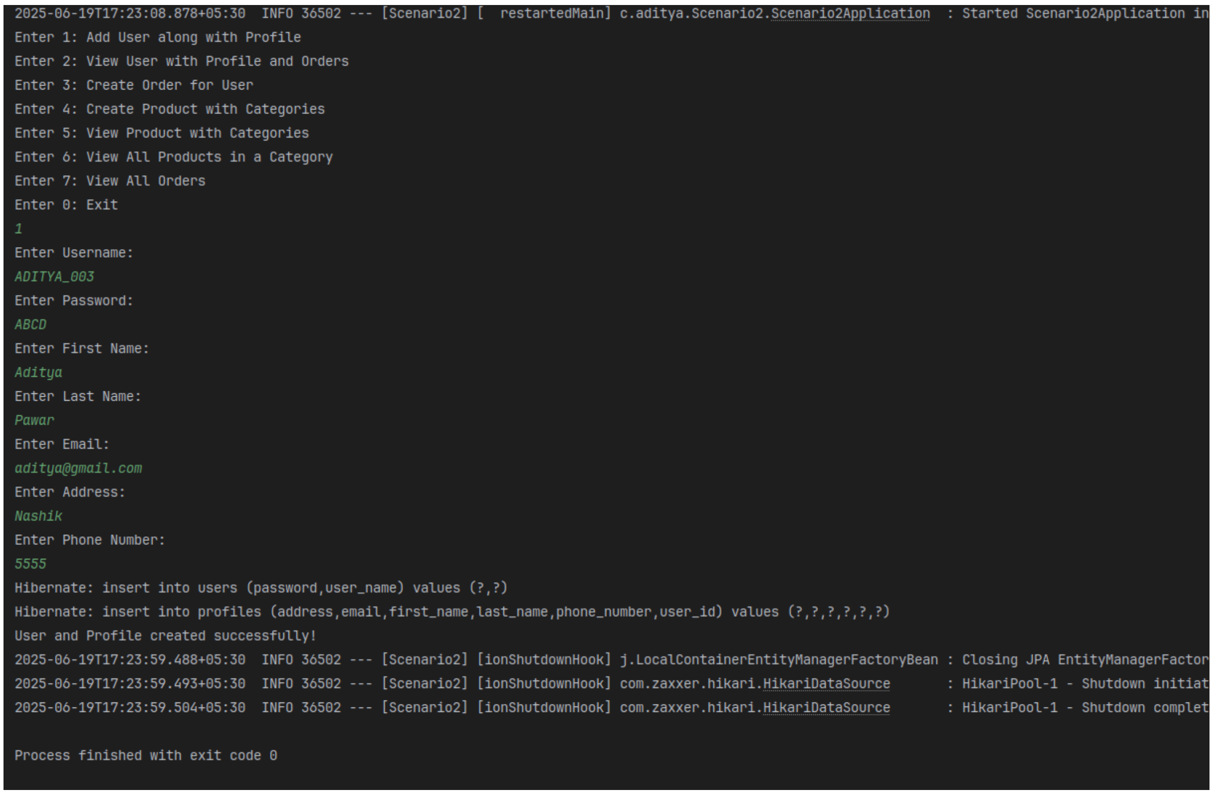
```
                System.out.println("Profile: " + foundUser.getProfile());
                System.out.println("Orders: " + foundUser.getOrders());
                System.out.println("*******************************************
            } else {
                System.out.println("User not found with ID: " + userId);
            }
            System.exit(0);
            break;

        case 3:
            System.out.println("Enter User ID to create an order:");
            int orderUserId = sc.nextInt();
            sc.nextLine(); // consume newline

            User orderUser = userService.getUserById(orderUserId);
            if (orderUser == null) {
                System.out.println("User not found with ID: " + orderUserId);
                System.exit(0);
            }

            System.out.println("Enter number of Products to add:");
            int productCount = sc.nextInt();
            sc.nextLine();

            List<Product> products = new ArrayList<>();

            for (int i = 1; i <= productCount; i++) {
                System.out.println("Enter Product Name for Product " + i + ":");
                String productName = sc.nextLine();

                System.out.println("Enter Product Price for Product " + i + ":");
                double productPrice = sc.nextDouble();
                sc.nextLine(); // consume leftover newline

                Product product = new Product(productName, productPrice);
                Product savedProduct = productService.returnProduct(product);

                products.add(savedProduct);
```

```
          }
          sc.nextLine();

          System.out.println("Enter Order Date (YYYY-MM-DD):");
          String orderDateStr = sc.nextLine();
          java.util.Date orderDate = java.sql.Date.valueOf(orderDateStr);

          // Create Order
          Order order = new Order(orderDate);
          order.setUser(orderUser);
          products = new ArrayList<>();
          for (Product product : products) {
             products.add(productService.returnProduct(product));
          }
          order.setProducts(products);

          orderService.addOrder(order);
          System.out.println("Order created successfully for User ID: "
          + orderUserId);
          break;

       case 4:
          System.out.println("Enter Product Name:");
          String productName = sc.nextLine();
          System.out.println("Enter Product Price:");
          double productPrice = sc.nextDouble();
          sc.nextLine(); // consume newline
          System.out.println("Enter number of Categories:");
          int categoryCount = sc.nextInt();
          sc.nextLine(); // consume newline

          List<String> categories = new ArrayList<>();
          for (int i = 1; i <= categoryCount; i++) {
             System.out.println("Enter Category Name for Category " + i + ":");
             String categoryName = sc.nextLine();
             categories.add(categoryName);
          }
          Product product = new Product(productName, productPrice);
```

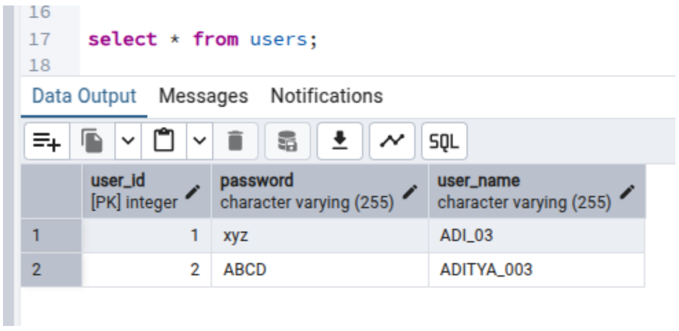
```
            List<Category> categoryList = new ArrayList<>();
            for (String categoryName : categories) {
                Category category = new Category(categoryName);
                categoryList.add(category);
            }
            product.setCategories(categoryList);
            productService.addProduct(product);
            System.out.println("Product created successfully with ID: "
            + product.getId());
            System.exit(0);
            break;

        case 5:
            System.out.println("Enter Product ID to view details:");
            int productId = sc.nextInt();
            sc.nextLine(); // consume newline
            Product foundProduct = productService.getProductById(productId);
            if (foundProduct != null) {
                System.out.println("********************************************
                System.out.println("Product Details:");
                System.out.println("Name: " + foundProduct.getName());
                System.out.println("Price: " + foundProduct.getPrice());
                System.out.println("Categories: " + foundProduct.getCategories()
                System.out.println("********************************************
            } else {
                System.out.println("Product not found with ID: " + productId);
            }
            System.exit(0);
            break;

        case 6:
            System.out.println("Enter Category ID:");
            int cid = sc.nextInt();
            sc.nextLine(); // consume newline

            List<Product> products1 = categoryService.getProductsInCategory(
            if (products1.isEmpty()) {
                System.out.println(" No products found in this category.");
            } else {
                System.out.println(" Products in category:");
                for (Product p : products1) {
                    System.out.println(p);
                }
            }
            System.exit(0);
            break;

        case 7:
            List<Order> orders = orderService.getAllOrders();
            if (orders.isEmpty()) {
                System.out.println("No orders found.");
            } else {
                System.out.println("*******************************************
                System.out.println("List of all Orders:");
                for (Order o : orders) {
                    System.out.println("Order ID: " + o.getId() + ", User: "
                    + o.getUser().getUserName() + ", Date: " + o.getOrderDate());
                    System.out.println("Products in Order:");
                    for (Product p : o.getProducts()) {
                        System.out.println("- " + p.getName() + " ($" + p.getPrice() +
                    }
                }
                System.out.println("*******************************************
            }
            System.exit(0);
            break;

        default:
            System.out.println("Invalid option. Please try again.");
            break;
    }
  }
}
```

Ouput:

CASE 1

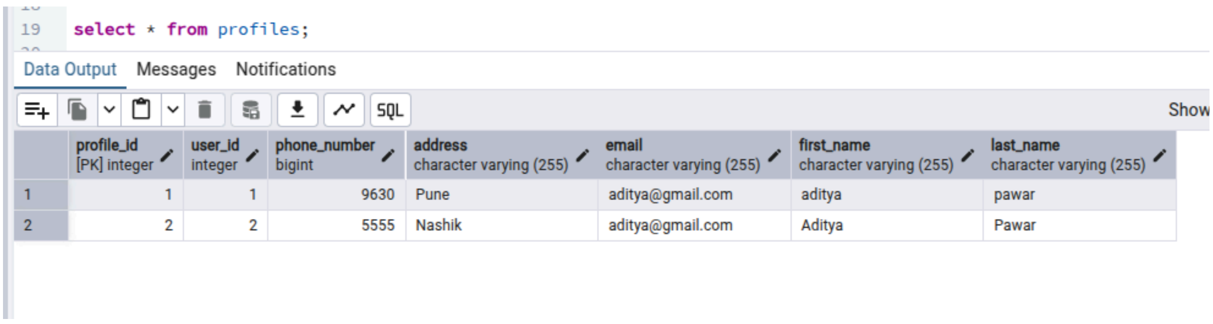


```
2025-06-19T17:23:08.878+05:30  INFO 36502 --- [Scenario2] [  restartedMain] c.aditya.Scenario2.Scenario2Application  : Started Scenario2Application in
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
1
Enter Username:
ADITYA_003
Enter Password:
ABCD
Enter First Name:
Aditya
Enter Last Name:
Pawar
Enter Email:
aditya@gmail.com
Enter Address:
Nashik
Enter Phone Number:
5555
Hibernate: insert into users (password,user_name) values (?,?)
Hibernate: insert into profiles (address,email,first_name,last_name,phone_number,user_id) values (?,?,?,?,?,?)
User and Profile created successfully!
2025-06-19T17:23:59.488+05:30  INFO 36502 --- [Scenario2] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactor
2025-06-19T17:23:59.493+05:30  INFO 36502 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiat
2025-06-19T17:23:59.504+05:30  INFO 36502 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown complet

Process finished with exit code 0
```



```sql
select * from users;
```

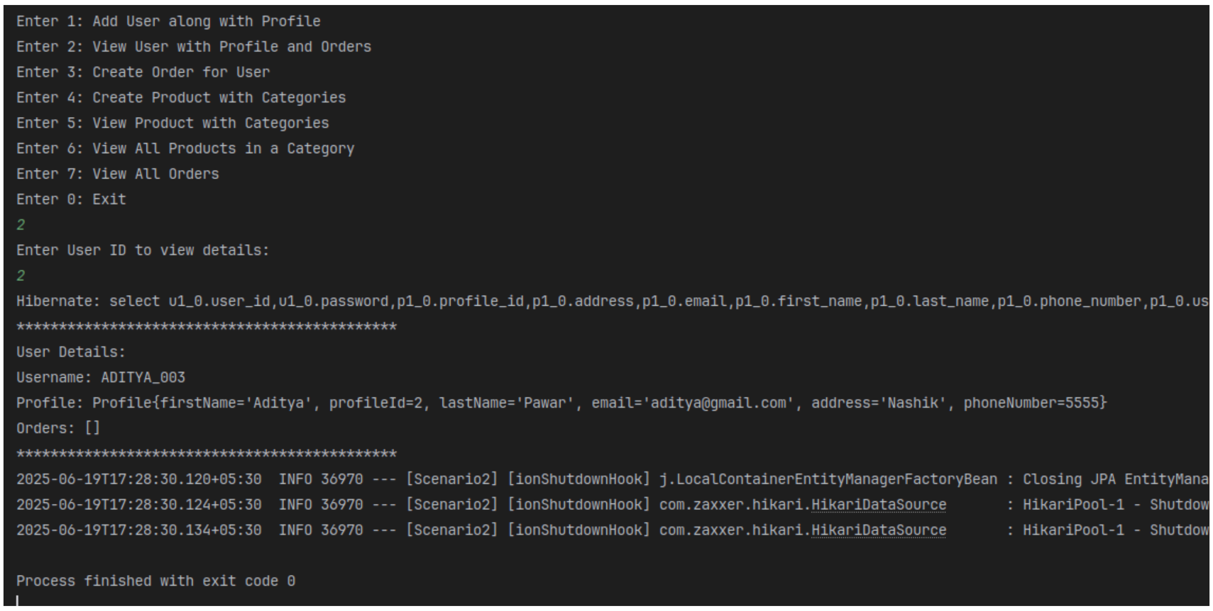
	user_id [PK] integer	password character varying (255)	user_name character varying (255)
1	1	xyz	ADI_03
2	2	ABCD	ADITYA_003



```sql
select * from profiles;
```

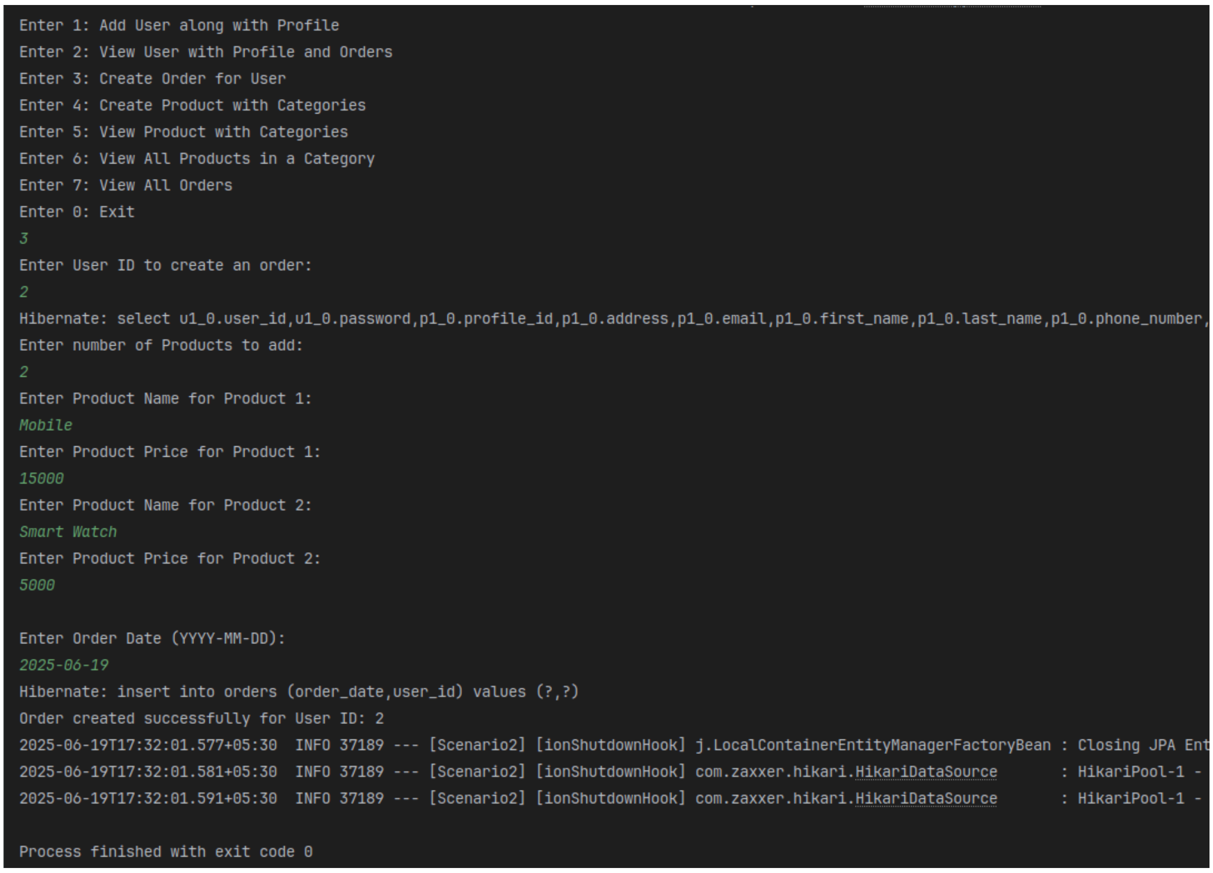
	profile_id [PK] integer	user_id integer	phone_number bigint	address character varying (255)	email character varying (255)	first_name character varying (255)	last_name character varying (255)
1	1	1	9630	Pune	aditya@gmail.com	aditya	pawar
2	2	2	5555	Nashik	aditya@gmail.com	Aditya	Pawar

CASE 2



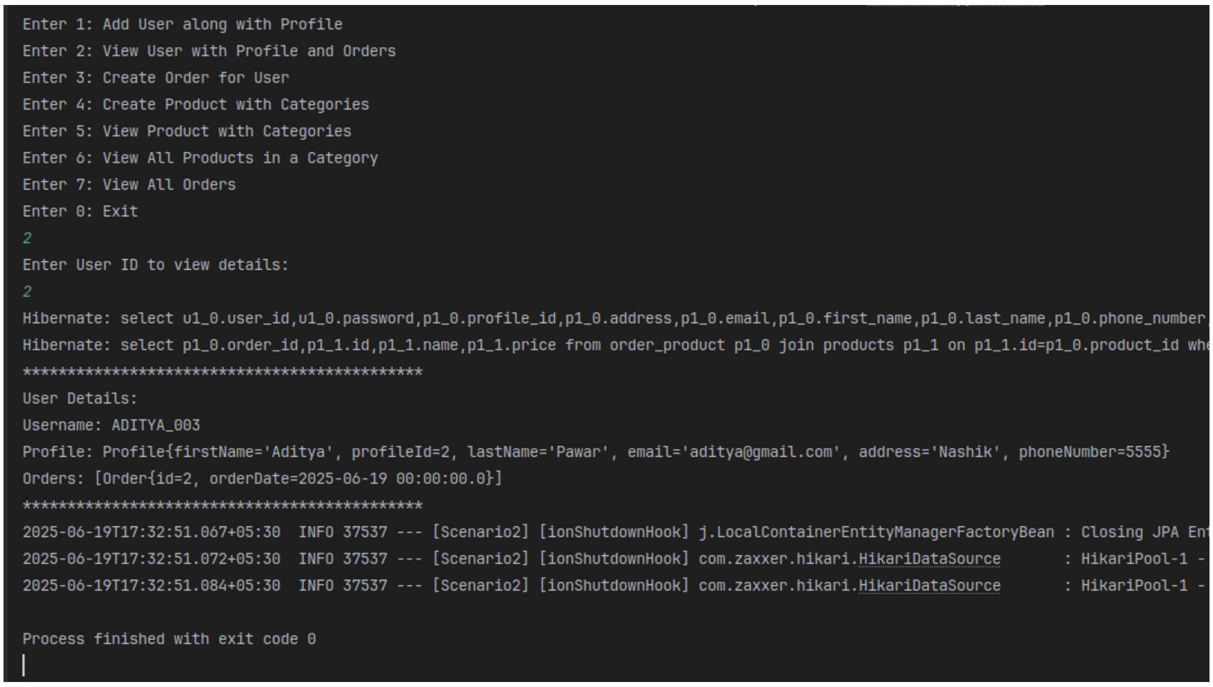
```
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
2
Enter User ID to view details:
2
Hibernate: select u1_0.user_id,u1_0.password,p1_0.profile_id,p1_0.address,p1_0.email,p1_0.first_name,p1_0.last_name,p1_0.phone_number,p1_0.us
*********************************************
User Details:
Username: ADITYA_003
Profile: Profile{firstName='Aditya', profileId=2, lastName='Pawar', email='aditya@gmail.com', address='Nashik', phoneNumber=5555}
Orders: []
*********************************************
2025-06-19T17:28:30.120+05:30  INFO 36970 --- [Scenario2] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMana
2025-06-19T17:28:30.124+05:30  INFO 36970 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdow
2025-06-19T17:28:30.134+05:30  INFO 36970 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdow

Process finished with exit code 0
```

CASE 3



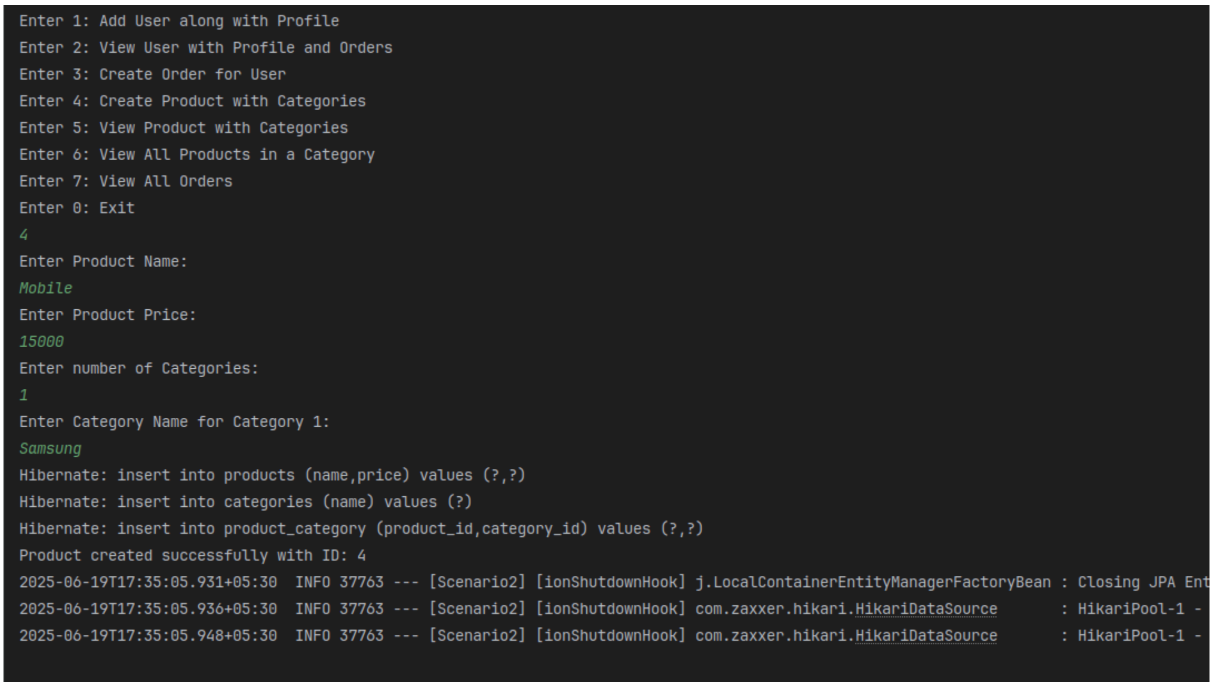
```
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
3
Enter User ID to create an order:
2
Hibernate: select u1_0.user_id,u1_0.password,p1_0.profile_id,p1_0.address,p1_0.email,p1_0.first_name,p1_0.last_name,p1_0.phone_number,
Enter number of Products to add:
2
Enter Product Name for Product 1:
Mobile
Enter Product Price for Product 1:
15000
Enter Product Name for Product 2:
Smart Watch
Enter Product Price for Product 2:
5000

Enter Order Date (YYYY-MM-DD):
2025-06-19
Hibernate: insert into orders (order_date,user_id) values (?,?)
Order created successfully for User ID: 2
2025-06-19T17:32:01.577+05:30  INFO 37189 --- [Scenario2] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2025-06-19T17:32:01.581+05:30  INFO 37189 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2025-06-19T17:32:01.591+05:30  INFO 37189 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -

Process finished with exit code 0
```

view orders

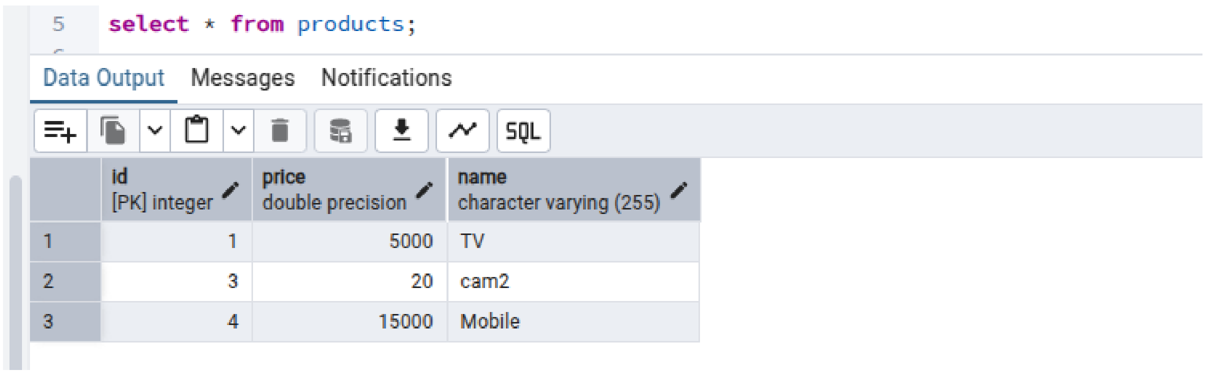


```
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
2
Enter User ID to view details:
2
Hibernate: select u1_0.user_id,u1_0.password,p1_0.profile_id,p1_0.address,p1_0.email,p1_0.first_name,p1_0.last_name,p1_0.phone_number,
Hibernate: select p1_0.order_id,p1_1.id,p1_1.name,p1_1.price from order_product p1_0 join products p1_1 on p1_1.id=p1_0.product_id whe
*********************************************
User Details:
Username: ADITYA_003
Profile: Profile{firstName='Aditya', profileId=2, lastName='Pawar', email='aditya@gmail.com', address='Nashik', phoneNumber=5555}
Orders: [Order{id=2, orderDate=2025-06-19 00:00:00.0}]
*********************************************
2025-06-19T17:32:51.067+05:30  INFO 37537 --- [Scenario2] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2025-06-19T17:32:51.072+05:30  INFO 37537 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2025-06-19T17:32:51.084+05:30  INFO 37537 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -

Process finished with exit code 0
```

CASE 4



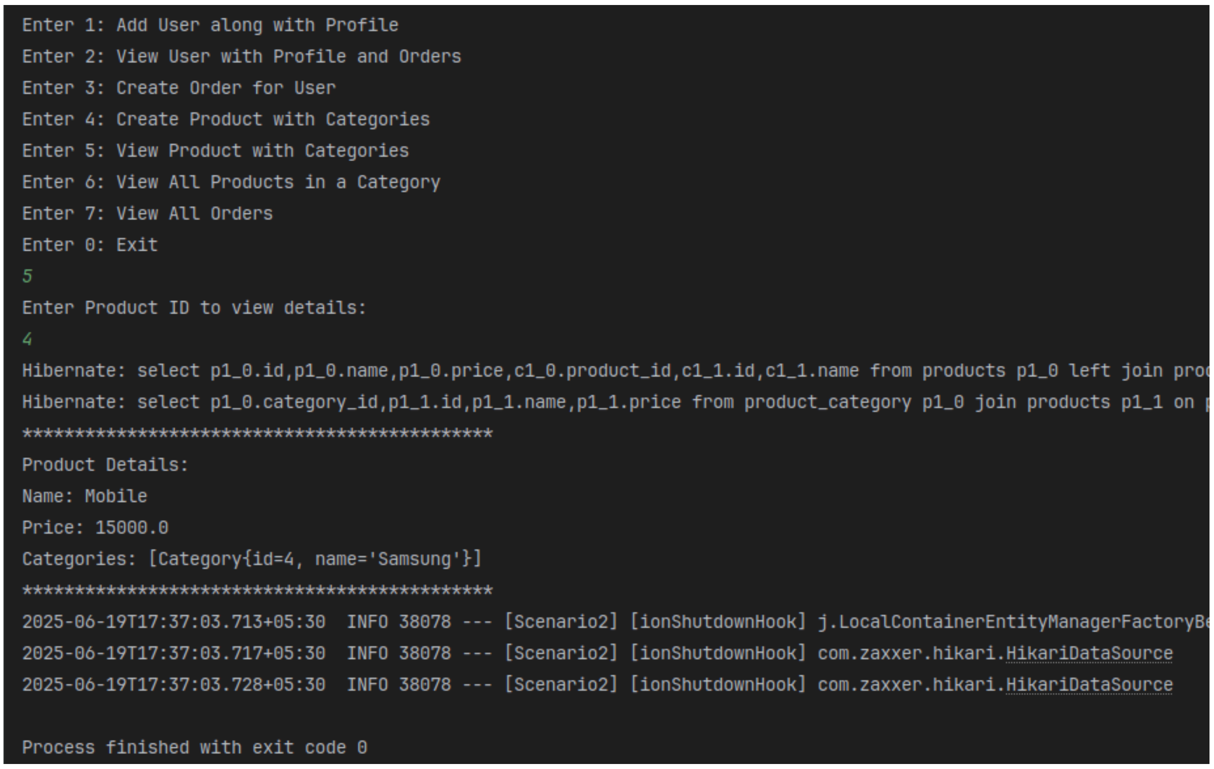
```
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
4
Enter Product Name:
Mobile
Enter Product Price:
15000
Enter number of Categories:
1
Enter Category Name for Category 1:
Samsung
Hibernate: insert into products (name,price) values (?,?)
Hibernate: insert into categories (name) values (?)
Hibernate: insert into product_category (product_id,category_id) values (?,?)
Product created successfully with ID: 4
2025-06-19T17:35:05.931+05:30  INFO 37763 --- [Scenario2] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Ent
2025-06-19T17:35:05.936+05:30  INFO 37763 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
2025-06-19T17:35:05.948+05:30  INFO 37763 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 -
```



```
select * from products;
```

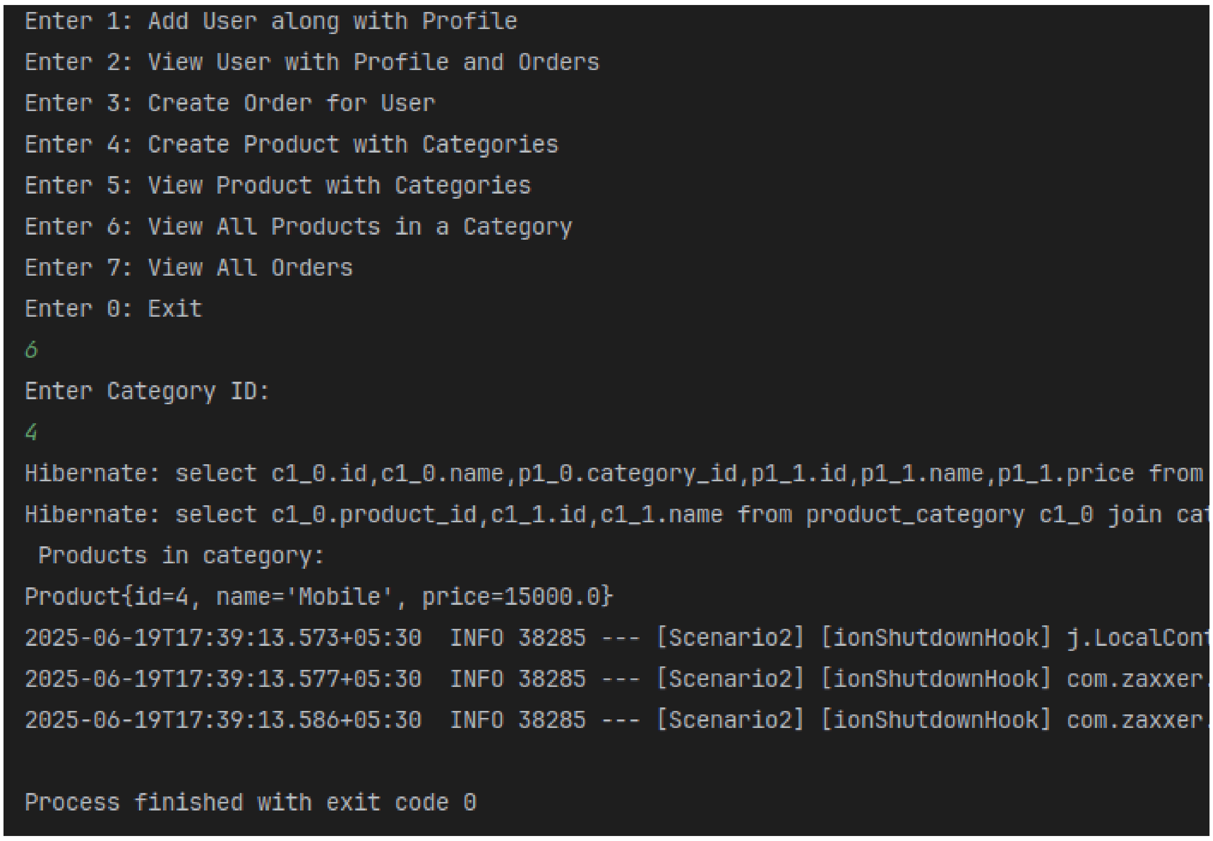
	id [PK] integer	price double precision	name character varying (255)
1	1	5000	TV
2	3	20	cam2
3	4	15000	Mobile

CASE 5

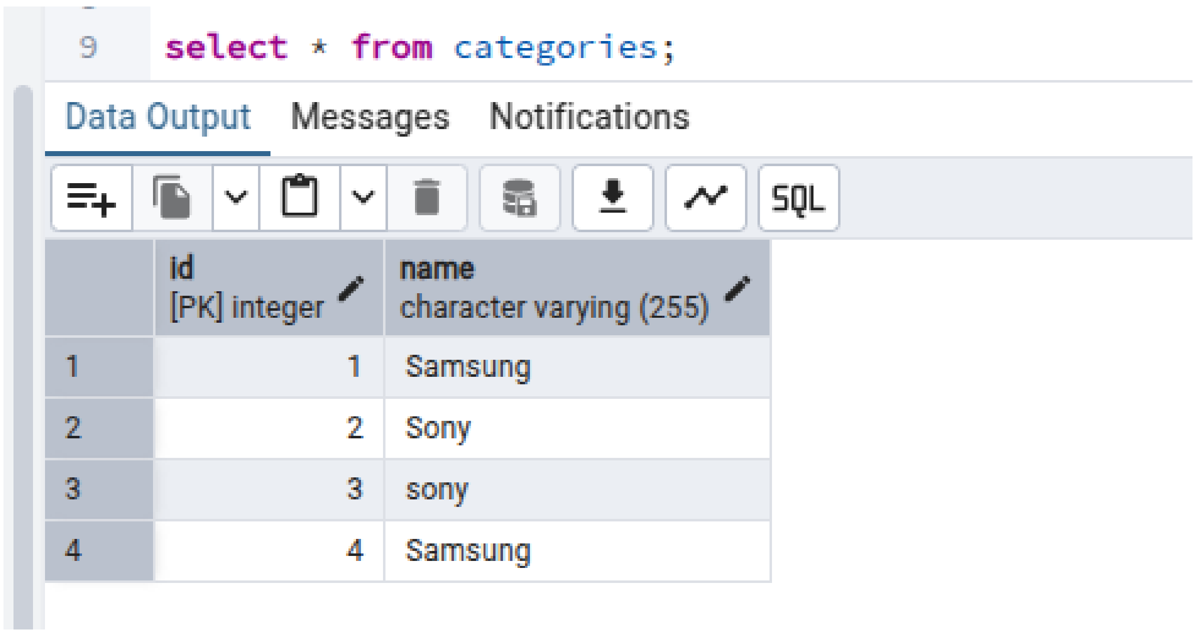


```
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
5
Enter Product ID to view details:
4
Hibernate: select p1_0.id,p1_0.name,p1_0.price,c1_0.product_id,c1_1.id,c1_1.name from products p1_0 left join prod
Hibernate: select p1_0.category_id,p1_1.id,p1_1.name,p1_1.price from product_category p1_0 join products p1_1 on p
*********************************************
Product Details:
Name: Mobile
Price: 15000.0
Categories: [Category{id=4, name='Samsung'}]
*********************************************
2025-06-19T17:37:03.713+05:30  INFO 38078 --- [Scenario2] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBe
2025-06-19T17:37:03.717+05:30  INFO 38078 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2025-06-19T17:37:03.728+05:30  INFO 38078 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0
```

CASE 6



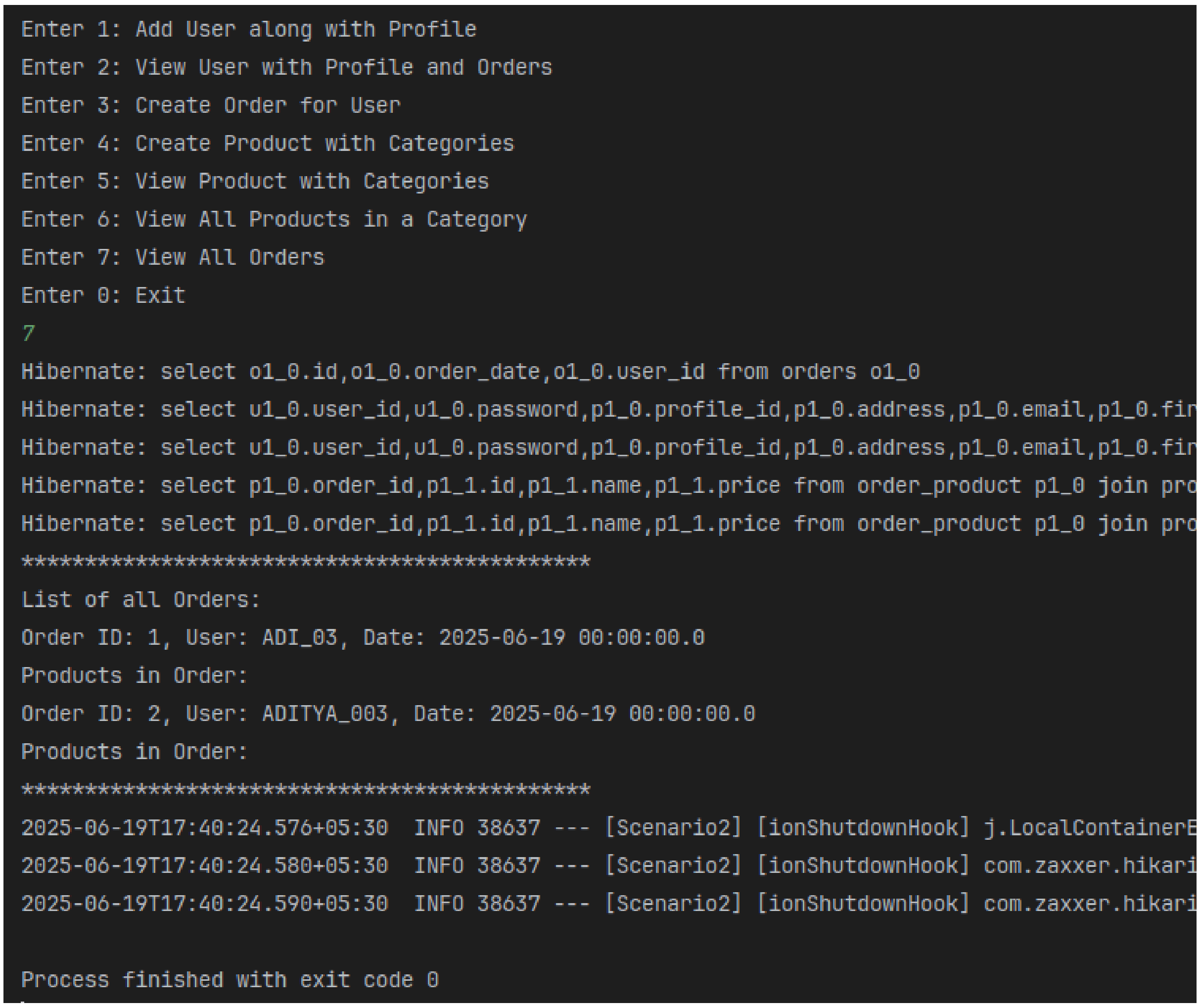
```
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
6
Enter Category ID:
4
Hibernate: select c1_0.id,c1_0.name,p1_0.category_id,p1_1.id,p1_1.name,p1_1.price from
Hibernate: select c1_0.product_id,c1_1.id,c1_1.name from product_category c1_0 join cat
 Products in category:
Product{id=4, name='Mobile', price=15000.0}
2025-06-19T17:39:13.573+05:30  INFO 38285 --- [Scenario2] [ionShutdownHook] j.LocalCont
2025-06-19T17:39:13.577+05:30  INFO 38285 --- [Scenario2] [ionShutdownHook] com.zaxxer.
2025-06-19T17:39:13.586+05:30  INFO 38285 --- [Scenario2] [ionShutdownHook] com.zaxxer.

Process finished with exit code 0
```



```
select * from categories;
```

	id [PK] integer	name character varying (255)
1	1	Samsung
2	2	Sony
3	3	sony
4	4	Samsung

CASE 7



```
Enter 1: Add User along with Profile
Enter 2: View User with Profile and Orders
Enter 3: Create Order for User
Enter 4: Create Product with Categories
Enter 5: View Product with Categories
Enter 6: View All Products in a Category
Enter 7: View All Orders
Enter 0: Exit
7
Hibernate: select o1_0.id,o1_0.order_date,o1_0.user_id from orders o1_0
Hibernate: select u1_0.user_id,u1_0.password,p1_0.profile_id,p1_0.address,p1_0.email,p1_0.fir
Hibernate: select u1_0.user_id,u1_0.password,p1_0.profile_id,p1_0.address,p1_0.email,p1_0.fir
Hibernate: select p1_0.order_id,p1_1.id,p1_1.name,p1_1.price from order_product p1_0 join pro
Hibernate: select p1_0.order_id,p1_1.id,p1_1.name,p1_1.price from order_product p1_0 join pro
*********************************************
List of all Orders:
Order ID: 1, User: ADI_03, Date: 2025-06-19 00:00:00.0
Products in Order:
Order ID: 2, User: ADITYA_003, Date: 2025-06-19 00:00:00.0
Products in Order:
*********************************************
2025-06-19T17:40:24.576+05:30  INFO 38637 --- [Scenario2] [ionShutdownHook] j.LocalContainerE
2025-06-19T17:40:24.580+05:30  INFO 38637 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari
2025-06-19T17:40:24.590+05:30  INFO 38637 --- [Scenario2] [ionShutdownHook] com.zaxxer.hikari

Process finished with exit code 0
```