Assignment No. 2

Name - Aditya Pawar

USN - 72233061J

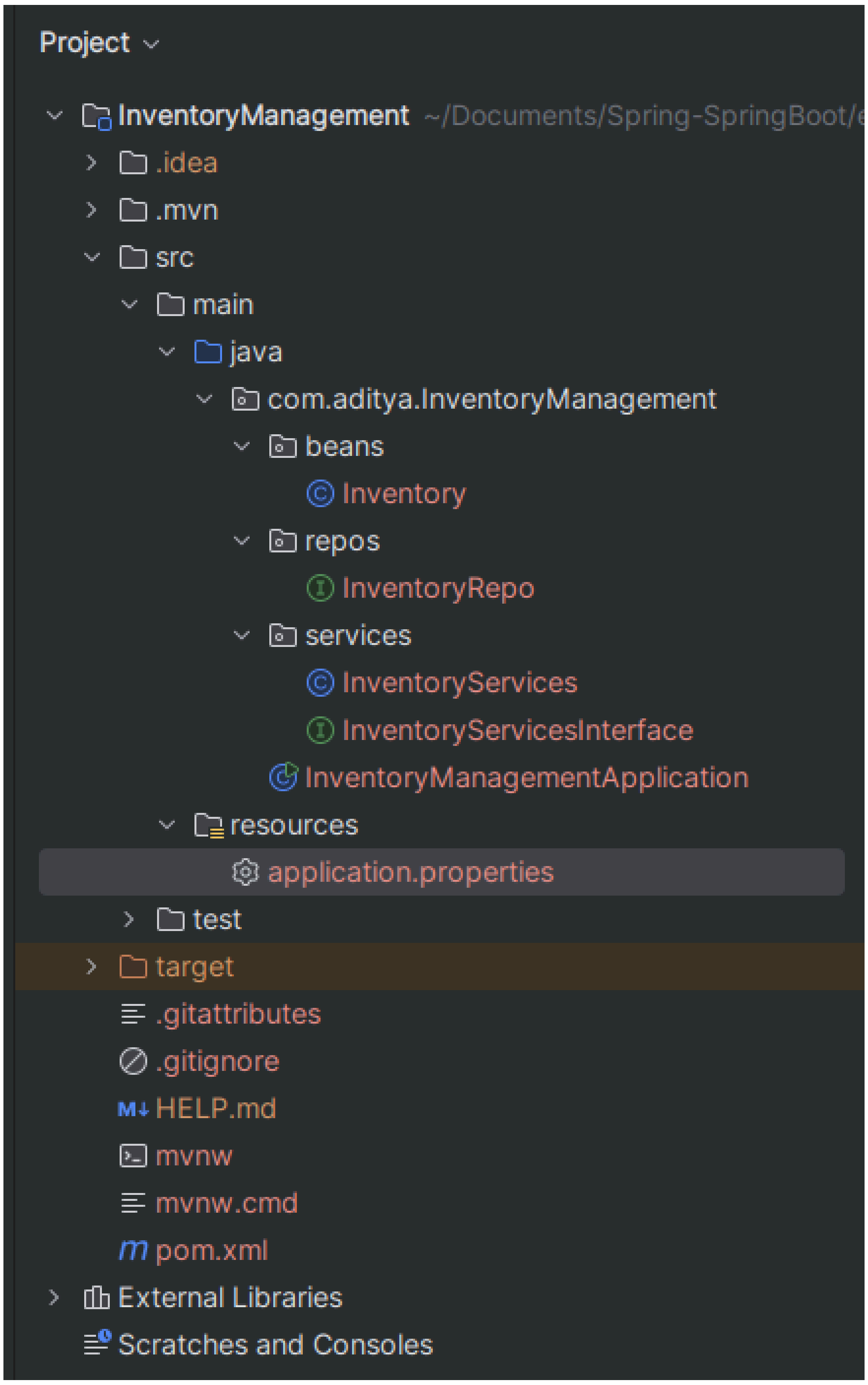
Que. 1 Inventory Management System

Problem Statement: Develop a RESTful API to manage the inventory of products in a warehouse. The system should allow users to:

- Create new product entries with details lüks name, description, price, and stock quantity.
- Read product information, including feticing details of a specific product by ID or listing all products.
- Update product details, such as modifying the price or updating stock levels.
- Delete products that are discontinued or obsolete.
- Fetch products by category, within a specified price range, and those that are currently in stock

---

Folder Structure:



```
InventoryManagement ~/Documents/Spring-SpringBoot/
  .idea
  .mvn
  src
    main
      java
        com.aditya.InventoryManagement
          beans
            Inventory
          repos
            InventoryRepo
          services
            InventoryServices
            InventoryServicesInterface
          InventoryManagementApplication
      resources
        application.properties
    test
  target
  .gitattributes
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

InventoryManagementApplication.java

```
package com.aditya.InventoryManagement;

import com.aditya.InventoryManagement.beans.Inventory;
import com.aditya.InventoryManagement.services.InventoryServices;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication
public class InventoryManagementApplication implements CommandLineRunn

    @Autowired
    InventoryServices inventoryServices;

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

    @Override
    public void run(String... args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter 1 to add new product");
        System.out.println("Enter 2 to read Product information by using product
        System.out.println("Enter 3 to update product details");
        System.out.println("Enter 4 to delete the products");
        System.out.println("Enter 5 to Fetch products");

        int operation = scanner.nextInt();

        switch (operation)
        {
            case 1:
                System.out.println("Enter Product Name");
                String productName = scanner.next();
                System.out.println("Enter product price");
                int price = scanner.nextInt();
                System.out.println("Enter stack Quantity");
                int stackQuantity = scanner.nextInt();
                scanner.nextLine();
                System.out.println("Enter Product Description");
                String description = scanner.nextLine();

                Inventory inventory1 = new Inventory();
                inventory1.setProductName(productName);
                inventory1.setDescription(description);
                inventory1.setPrice(price);
                inventory1.setStockQuantity(stackQuantity);
                inventoryServices.addProduct(inventory1);
                break;

            case 2:
                System.out.println("Enter Product Id");
                int productId = scanner.nextInt();
                try{
                    Optional<Inventory> optional = inventoryServices.findProductByI
                    productId);
                    Inventory inventory2 = optional.get();
                    System.out.println(inventory2.toString());
                }
                catch (Exception e){
                    System.err.println("Id not Found"+e);
                }
                break;

            case 3:
                System.out.println("Enter product Id to be updated");
                productId = scanner.nextInt();
                inventoryServices.updateInventory(productId);
                break;

            case 4:
                System.out.println("Enter product Id to be updated");
                productId = scanner.nextInt();
                inventoryServices.deleteProduct(productId);
                break;

            case 5:
                System.out.println("Fetching all products information");
                List<Inventory> productlist = inventoryServices.getAllProducts();
                productlist.forEach(inventory → System.out.println(inventory));
                break;

            default:
                System.out.println("Please enter the Valid Operation... Thank You!!!"
          }
        }
    }
```

Inventory.java

```
package com.aditya.InventoryManagement.beans;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@NoArgsConstructor
@Entity
@Table(name = "inventoryTable")
public class Inventory {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private @Getter @Setter int productId;
   @NotBlank(message = "Product Name cannot be Empty")
   private @Getter @Setter String productName;
//    @Column(length = 1000)
//    @Lob
   @Column(name = "description", columnDefinition = "TEXT")
   private @Getter @Setter String description;
   @NotNull(message = "price cannot be null")
```

```
    private @Getter @Setter int price;
    private @Getter @Setter int stockQuantity;

    public Inventory(int productId, int stockQuantity, int price, String
    description, String productName) {
        this.productId = productId;
        this.stockQuantity = stockQuantity;
        this.price = price;
        this.description = description;
        this.productName = productName;
    }

    @Override
    public String toString() {
        return "Inventory{" +
                "productId=" + productId +
                ", productName='" + productName + '\'' +
                ", description='" + description + '\'' +
                ", price=" + price +
                ", stockQuantity=" + stockQuantity +
                '}';
    }
}
```

InventroyService.java

```
package com.aditya.InventoryManagement.services;

import com.aditya.InventoryManagement.beans.Inventory;
import com.aditya.InventoryManagement.repos.InventoryRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;
import java.util.Scanner;
```

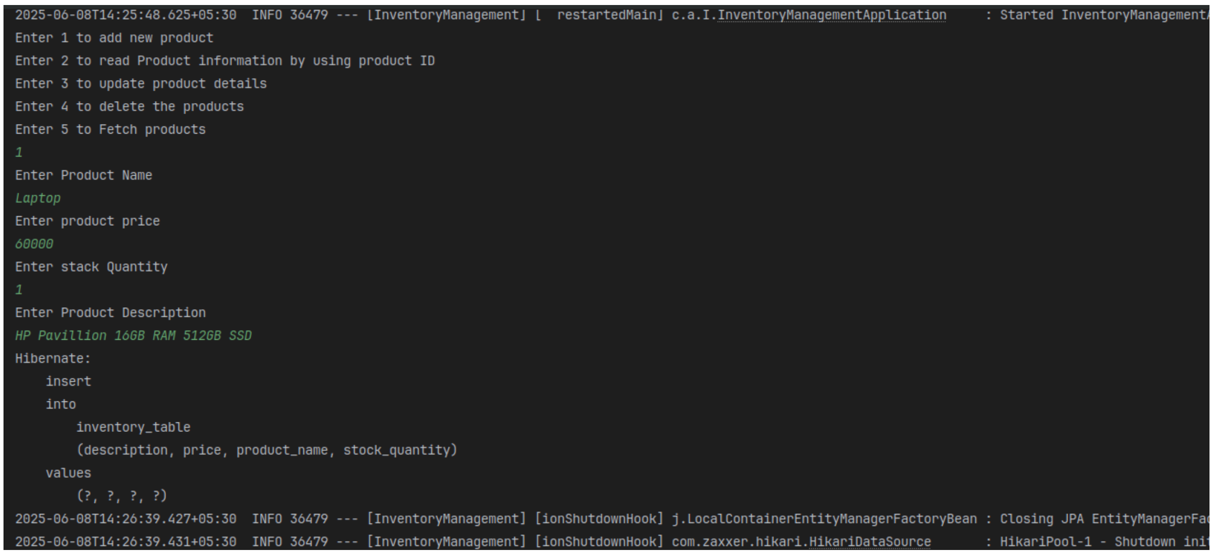
```
@Service
public class InventoryServices implements InventoryServicesInterface {

    @Autowired
    InventoryRepo inventoryRepo;

    @Override
    public void addProduct(Inventory inventory) {
        inventoryRepo.save(inventory);
    }

    @Override
    public Optional<Inventory> findProductById(int productId) {
        return inventoryRepo.findById(productId);
    }

    @Override
    public void updateInventory(int productId) {

        Optional<Inventory> optional = inventoryRepo.findById(productId);
        Inventory inventory = optional.get();
        try
        {
            if (inventory != null){
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter New Product Name");
                String newProductName = sc.next();
                System.out.println("Enter New Product Price");
                int newPrice = sc.nextInt();
                System.out.println("Update Stock Levels");
                int newStockQuantity = sc.nextInt();
                inventory.setProductName(newProductName);
                inventory.setStockQuantity(newStockQuantity);
                inventory.setPrice(newPrice);
                inventoryRepo.save(inventory);
            }
        }
        catch (Exception e){
            System.err.println("Id not found "+e);
        }
    }

    @Override
    public void deleteProduct(int productId) {
        try
        {
            inventoryRepo.deleteById(productId);
            System.out.println("Successfully deleted product id = "+productId);
        }
        catch (Exception e) {
            System.err.println("Product Id "+productId+" not found"+e);
        }
    }

    @Override
    public List<Inventory> getAllProducts() {
        return inventoryRepo.findAll();
    }
}
```

InventoryServicesInterface.java

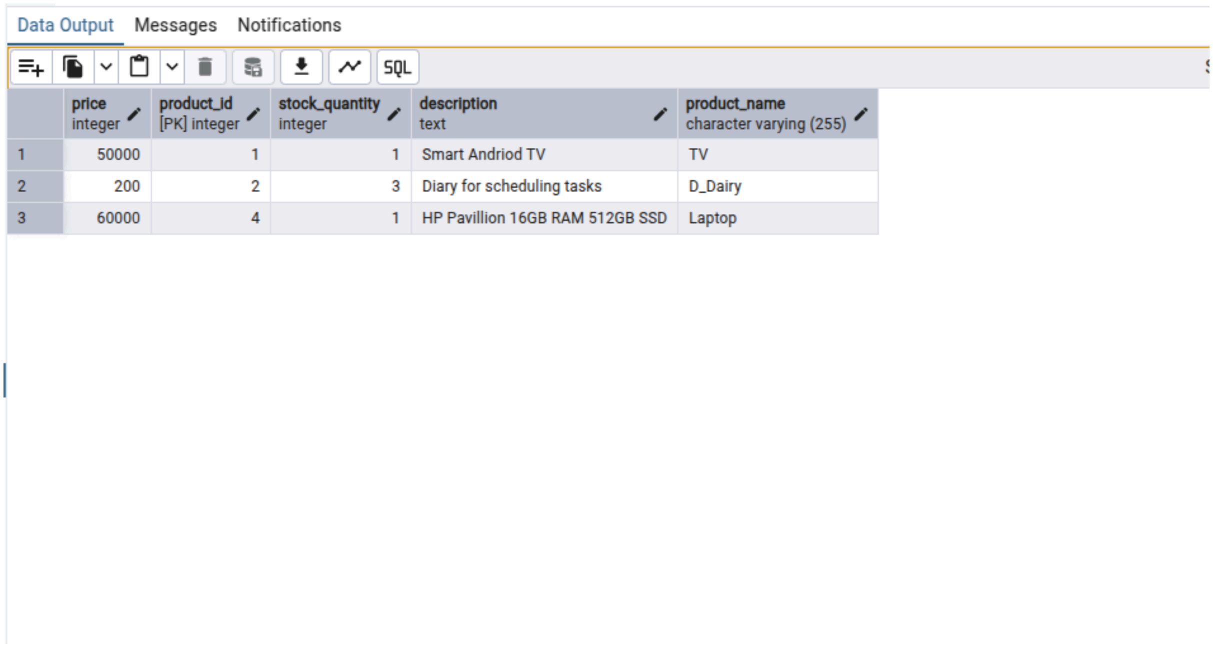
```
package com.aditya.InventoryManagement.services;

import com.aditya.InventoryManagement.beans.Inventory;

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

public interface InventoryServicesInterface {

    public void addProduct(Inventory inventory);

    public Optional<Inventory> findProductById(int productId);

    public void updateInventory(int productId);

    public void deleteProduct(int productId);

    public List<Inventory> getAllProducts();

}
```

Output:

case 1—>

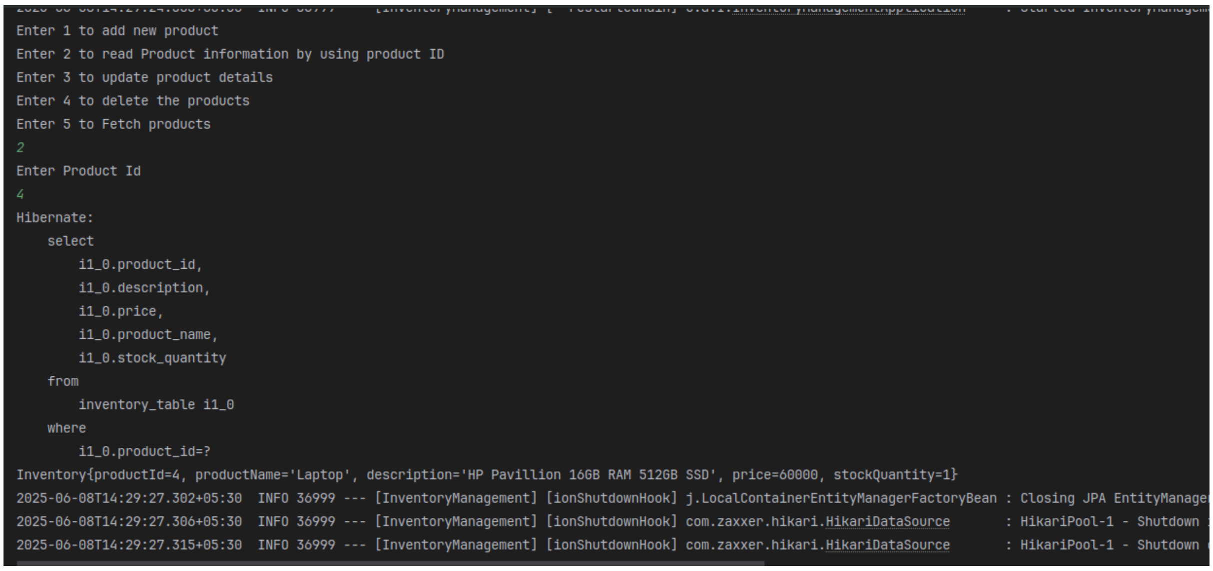


```
2025-06-08T14:25:48.625+05:30  INFO 36479 --- [InventoryManagement] [  restartedMain] c.a.I.InventoryManagementApplication     : Started InventoryManagement
Enter 1 to add new product
Enter 2 to read Product information by using product ID
Enter 3 to update product details
Enter 4 to delete the products
Enter 5 to Fetch products
1
Enter Product Name
Laptop
Enter product price
60000
Enter stack Quantity
1
Enter Product Description
HP Pavillion 16GB RAM 512GB SSD
Hibernate:
    insert
    into
        inventory_table
        (description, price, product_name, stock_quantity)
    values
        (?, ?, ?, ?)
2025-06-08T14:26:39.427+05:30  INFO 36479 --- [InventoryManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFa
2025-06-08T14:26:39.431+05:30  INFO 36479 --- [InventoryManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown ini
```



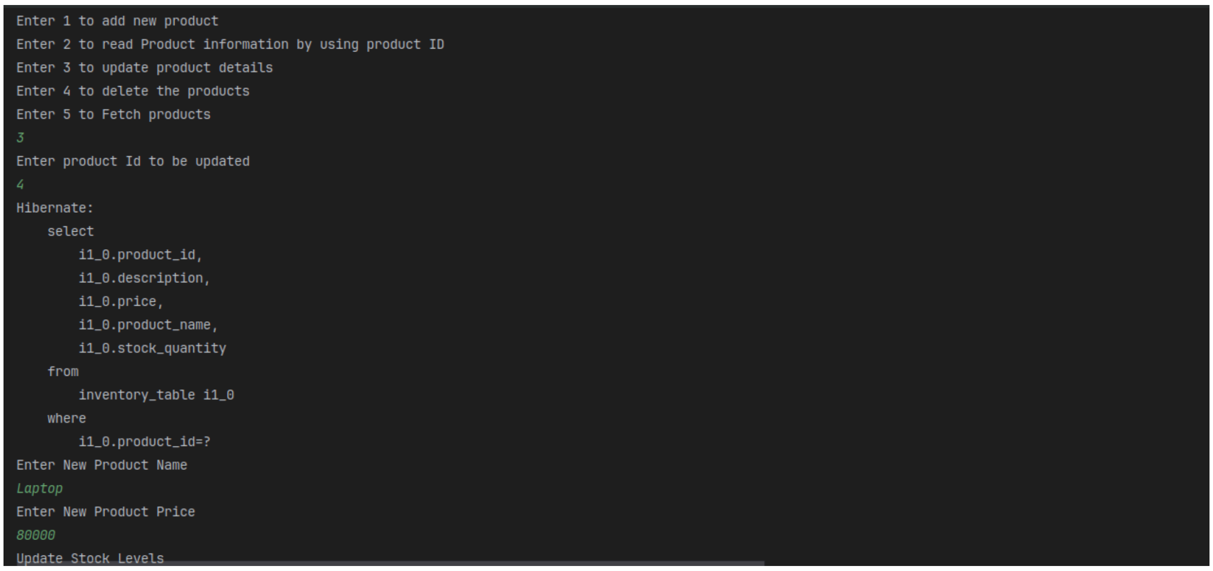
	price integer	product_id [PK] integer	stock_quantity integer	description text	product_name character varying (255)
1	50000	1	1	Smart Andriod TV	TV
2	200	2	3	Diary for scheduling tasks	D_Dairy
3	60000	4	1	HP Pavillion 16GB RAM 512GB SSD	Laptop

case 2—>

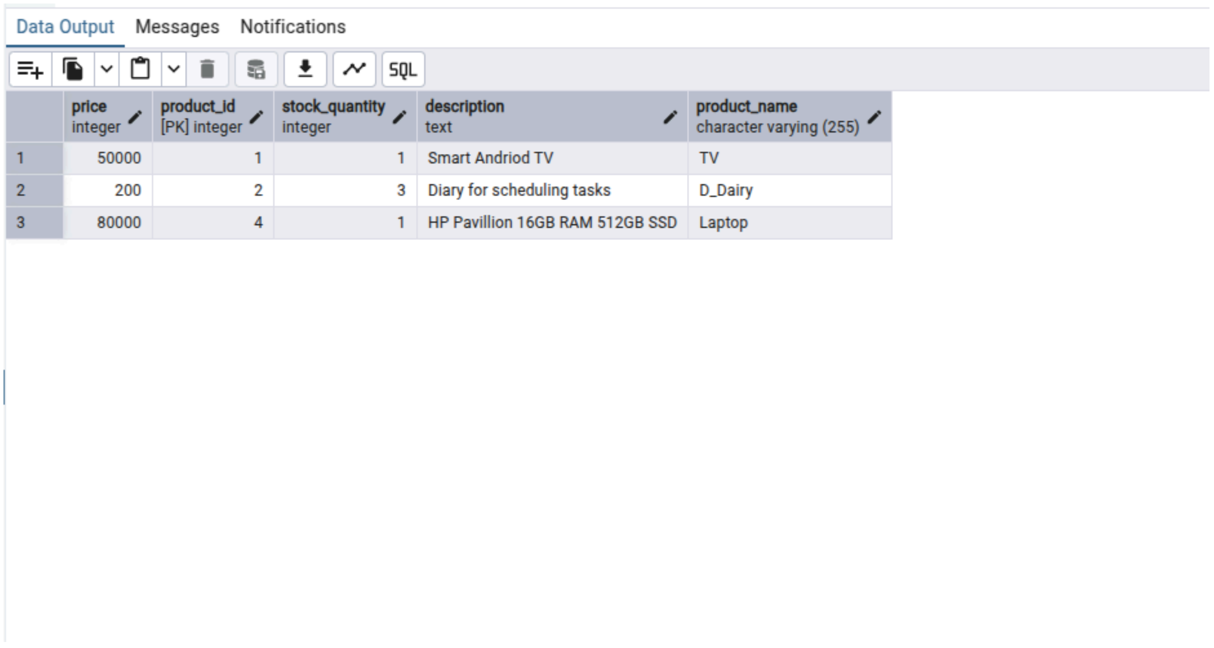


```
Enter 1 to add new product
Enter 2 to read Product information by using product ID
Enter 3 to update product details
Enter 4 to delete the products
Enter 5 to Fetch products
2
Enter Product Id
4
Hibernate: 
    select
        i1_0.product_id,
        i1_0.description,
        i1_0.price,
        i1_0.product_name,
        i1_0.stock_quantity 
    from
        inventory_table i1_0 
    where
        i1_0.product_id=?
Inventory{productId=4, productName='Laptop', description='HP Pavillion 16GB RAM 512GB SSD', price=60000, stockQuantity=1}
2025-06-08T14:29:27.302+05:30  INFO 36999 --- [InventoryManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManage
2025-06-08T14:29:27.306+05:30  INFO 36999 --- [InventoryManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown
2025-06-08T14:29:27.315+05:30  INFO 36999 --- [InventoryManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown
```

case 3—>



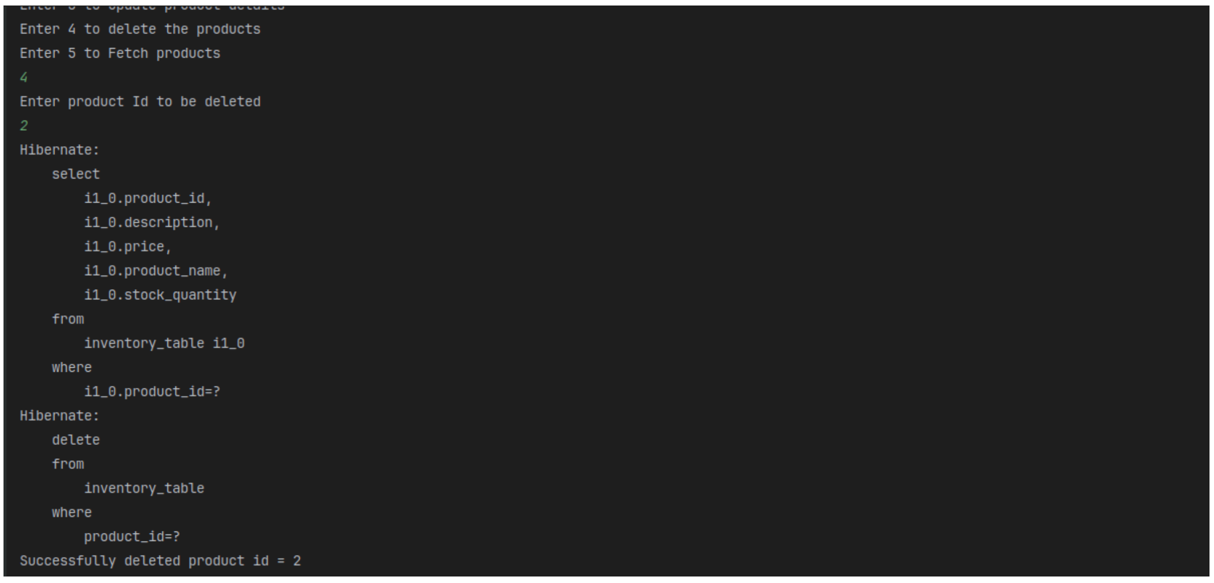
```
Enter 1 to add new product
Enter 2 to read Product information by using product ID
Enter 3 to update product details
Enter 4 to delete the products
Enter 5 to Fetch products
3
Enter product Id to be updated
4
Hibernate: 
    select
        i1_0.product_id,
        i1_0.description,
        i1_0.price,
        i1_0.product_name,
        i1_0.stock_quantity 
    from
        inventory_table i1_0 
    where
        i1_0.product_id=?
Enter New Product Name
Laptop
Enter New Product Price
80000
Update Stock Levels
```



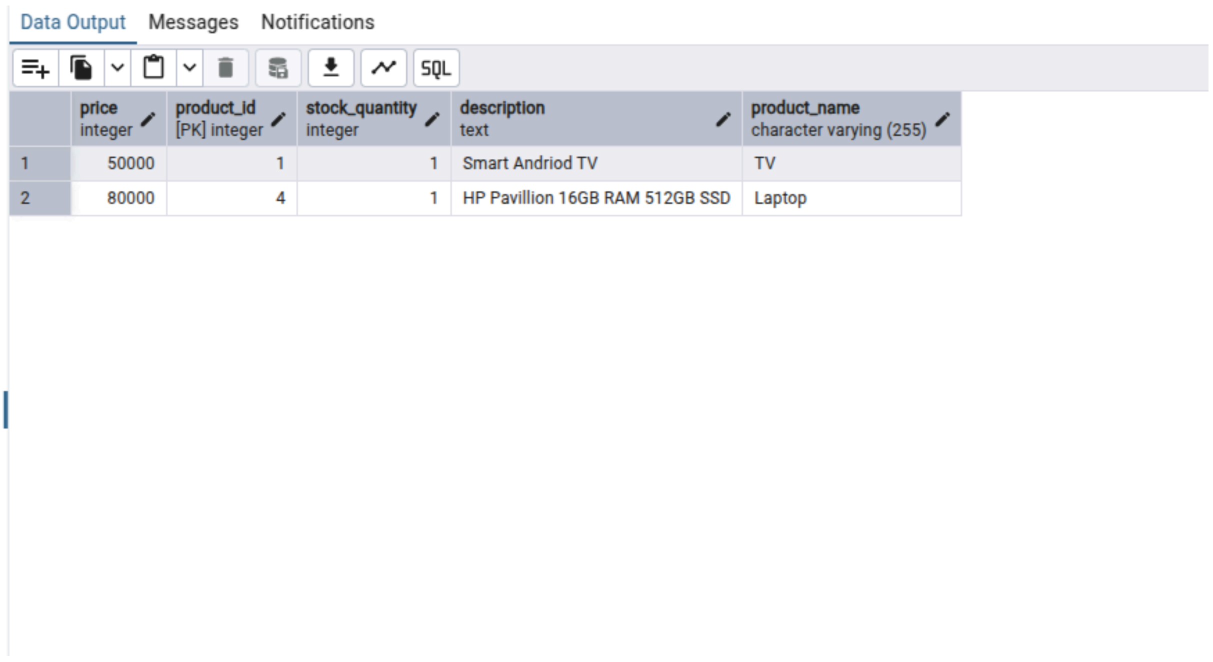
Data Output | Messages | Notifications

	price integer	product_id [PK] integer	stock_quantity integer	description text	product_name character varying (255)
1	50000	1	1	Smart Andriod TV	TV
2	200	2	3	Diary for scheduling tasks	D_Dairy
3	80000	4	1	HP Pavillion 16GB RAM 512GB SSD	Laptop

case4—>



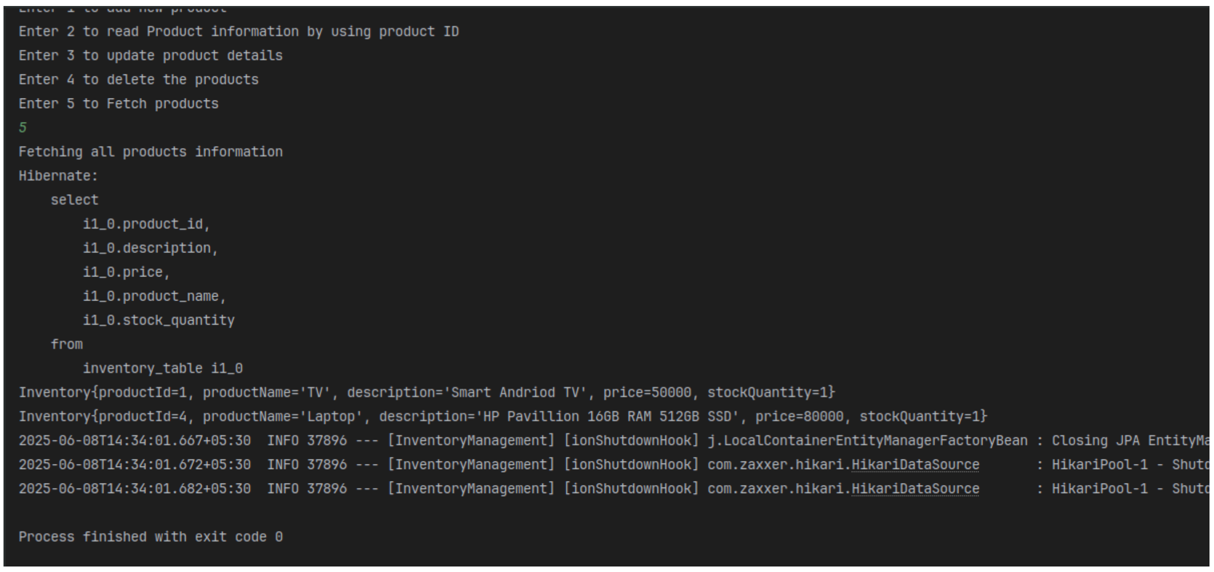
```
Enter 4 to delete the products
Enter 5 to Fetch products
4
Enter product Id to be deleted
2
Hibernate:
    select
        i1_0.product_id,
        i1_0.description,
        i1_0.price,
        i1_0.product_name,
        i1_0.stock_quantity
    from
        inventory_table i1_0
    where
        i1_0.product_id=?
Hibernate:
    delete
    from
        inventory_table
    where
        product_id=?
Successfully deleted product id = 2
```



Data Output Messages Notifications

	price integer	product_id [PK] integer	stock_quantity integer	description text	product_name character varying (255)
1	50000	1	1	Smart Andriod TV	TV
2	80000	4	1	HP Pavillion 16GB RAM 512GB SSD	Laptop

case 5—>



```
Enter 2 to read Product information by using product ID
Enter 3 to update product details
Enter 4 to delete the products
Enter 5 to Fetch products
5
Fetching all products information
Hibernate:
    select
        i1_0.product_id,
        i1_0.description,
        i1_0.price,
        i1_0.product_name,
        i1_0.stock_quantity
    from
        inventory_table i1_0
Inventory{productId=1, productName='TV', description='Smart Andriod TV', price=50000, stockQuantity=1}
Inventory{productId=4, productName='Laptop', description='HP Pavillion 16GB RAM 512GB SSD', price=80000, stockQuantity=1}
2025-06-08T14:34:01.667+05:30  INFO 37896 --- [InventoryManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMa
2025-06-08T14:34:01.672+05:30  INFO 37896 --- [InventoryManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutd
2025-06-08T14:34:01.682+05:30  INFO 37896 --- [InventoryManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutd

Process finished with exit code 0
```

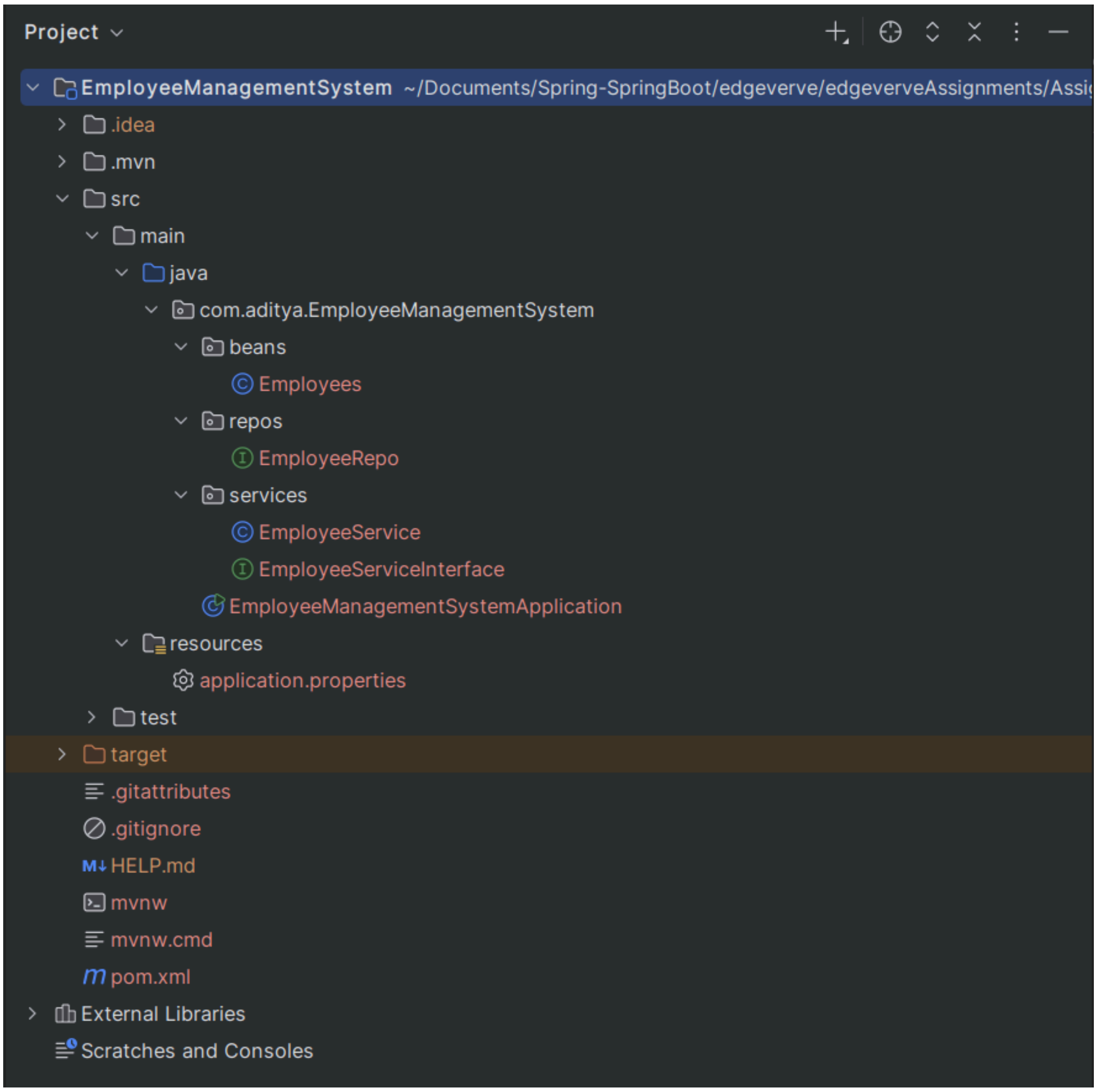
Que. 2 Employee Management System

Problem Statement: Create an application to manage employee records in an organization. The system should support:

- Create operations to add new employees with attributes like name, role, department, and salary
- Read functionalities to retrieve employee details by ID or list all employees.

- Update capabilities to modify employee information, such as role changes or salary adjustments.
- Delete operations to remove employees who have left the organization.
- err.println("Invalid Option",e.getMessage());

Folder Structure:



```
Project
EmployeeManagementSystem ~/Documents/Spring-SpringBoot/edgeverve/edgeverveAssignments/Assi
  .idea
  .mvn
  src
    main
      java
        com.aditya.EmployeeManagementSystem
          beans
            Employees
          repos
            EmployeeRepo
          services
            EmployeeService
            EmployeeServiceInterface
          EmployeeManagementSystemApplication
      resources
        application.properties
    test
  target
  .gitattributes
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

EmployeeManagementSystemApplication.java

```
package com.aditya.EmployeeManagementSystem;
```

```
import com.aditya.EmployeeManagementSystem.beans.Employees;
import com.aditya.EmployeeManagementSystem.services.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.ArrayList;
import java.util.List;
import java.util.Scanner;

@SpringBootApplication
public class EmployeeManagementSystemApplication implements CommandL

    @Autowired
    EmployeeService employeeService;

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

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

        System.out.println("Enter 1 to add new Employee");
        System.out.println("Enter 2 to find Employee by using ID");
        System.out.println("Enter 3 to list all the Employees");
        System.out.println("Enter 4 to updated Employee Details");
        System.out.println("Enter 5 to delete particular Employee");
        System.out.println("Enter 6 to fetch employees by data");

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

        switch (operations)
        {
            case 1:
```

```
        System.out.println("Enter Employee Name");
        String empName = sc.nextLine();

        System.out.println("Enter Employee Role");
        String empRole = sc.nextLine();

        System.out.println("Enter Employee Department");
        String empDept = sc.nextLine();

        System.out.println("Enter Employee Salary");
        double empSalary  = sc.nextDouble();
        sc.nextLine();

        System.out.println("Enter Employee Location");
        String empLoc = sc.nextLine();

        Employees employees = new Employees();
        employees.setName(empName);
        employees.setRole(empRole);
        employees.setDepartment(empDept);
        employees.setSalary(empSalary);
        employees.setLocation(empLoc);
        employeeService.addEmployee(employees);
        break;

    case 2:
        System.out.println("Enter Employee ID");
        int empId = sc.nextInt();
        try {
            Employees employees1 = employeeService.getEmployeeById(emp
            System.out.println(employees1.toString());
        } catch (Exception e) {
            System.err.println("Id not valid"+e);
        }
        break;

    case 3:
        System.out.println("List of All Employees");
        List<Employees> employeesList = employeeService.getAllEmployee
        employeesList.forEach(employees2 → System.out.println(employees
        break;

    case 4:
        System.out.println("Enter Employee ID tp be updated");
        empId = sc.nextInt();
        sc.nextLine();
        try {
            Employees existingEmployee = employeeService.getEmployeeByI
            if (existingEmployee == null){
                System.out.println("Employee with ID"+empId+" Not Found!!!");
                break;
            }

            System.out.println("update employee Name");
            String newEmpName = sc.nextLine();
            System.out.println("update employee role");
            String newEmpRole = sc.nextLine();
            System.out.println("update employee department");
            String newEmpDept = sc.nextLine();
            System.out.println("update employee salary");
            double newEmpSalary = sc.nextDouble();
            sc.nextLine();
            System.out.println("update employee location");
            String newEmpLoc = sc.nextLine();


            existingEmployee.setName(newEmpName);
            existingEmployee.setRole(newEmpRole);
            existingEmployee.setDepartment(newEmpDept);
            existingEmployee.setSalary(newEmpSalary);
            existingEmployee.setLocation(newEmpLoc);
            employeeService.updateEmployee(existingEmployee);
            System.out.println("Employee is updated Successfully...");
        } catch (Exception e) {
            System.err.println("ID Not Valid"+e.getMessage());
        }
```

```
          break;

        case 5:
          System.out.println("Enter the Employee ID, Who is no longer working
          organization");
          empId = sc.nextInt();
          sc.nextLine();
          employeeService.deleteEmploye(empId);
          break;

        case 6:
          System.out.println("Search by:");
          System.out.println("1: Department");
          System.out.println("2: Role");
          System.out.println("3: Location");

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

          try {
            List<Employees> filteredList = new ArrayList<>();

            switch (option) {
              case 1:
                System.out.println("Enter Department");
                String dept = sc.nextLine();
                filteredList = employeeService.filterByDepartment(dept);
                break;
              case 2:
                System.out.println("Enter Role");
                String role = sc.nextLine();
                filteredList = employeeService.filterByRole(role);
                break;
              case 3:
                System.out.println("Enter Location");
                String loc = sc.nextLine();
                filteredList = employeeService.filterByLocation(loc);
                break;
```

```
                    default:
                        System.out.println("Select Proper option...");
                }

                if (!filteredList.isEmpty()) {
                    for (Employees e : filteredList) {
                        System.out.println(e);
                    }
                } else {
                    System.out.println("No Matching Record Found!!!");
                }
            } catch (Exception e) {
                System.err.println("Invalid Option"+e.getMessage());
            }
            break;

        default:
            System.out.println("Invalid Operation Input!!!");
    }
  }
}
```

Employees.java

```
package com.aditya.EmployeeManagementSystem.beans;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@NoArgsConstructor
@Entity
@Table(name = "employees")
public class Employees {
```

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

@NotBlank(message = "Name is required")
private @Getter @Setter String name;

@NotBlank(message = "Role is required")
private @Getter @Setter String role;

@NotBlank(message = "Department is required")
private @Getter @Setter String department;

@NotNull(message = "Salary is mandatory")
private @Getter @Setter double salary;

@NotBlank(message = "Location is required")
private @Getter @Setter String location;

public Employees(int employeeId, double salary, String location, String
name, String role, String department) {
    this.employeeId = employeeId;
    this.salary = salary;
    this.location = location;
    this.name = name;
    this.role = role;
    this.department = department;
}

@Override
public String toString() {
    return "Employees{" +
            "employeeId=" + employeeId +
            ", name='" + name + '\'' +
            ", role='" + role + '\'' +
            ", department='" + department + '\'' +
            ", salary=" + salary +
            ", location='" + location + '\'' +
            '}';
    }
}
```

EmployeeService.java

```
package com.aditya.EmployeeManagementSystem.services;

import com.aditya.EmployeeManagementSystem.beans.Employees;
import com.aditya.EmployeeManagementSystem.repos.EmployeeRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeService implements EmployeeServiceInterface {

    @Autowired
    EmployeeRepo employeeRepo;

    @Override
    public void addEmployee(Employees employees) {
        employeeRepo.save(employees);
    }

    @Override
    public Employees getEmployeeById(int employeeId) {
        return employeeRepo.findById(employeeId).orElse(null);
    }

    @Override
    public List<Employees> getAllEmployees() {
        return employeeRepo.findAll();
    }
    @Override
    public void updateEmployee(Employees employees) {
        employeeRepo.save(employees);
    }

    @Override
    public void deleteEmploye(int empId) {
        employeeRepo.deleteById(empId);
    }

    @Override
    public List<Employees> filterByDepartment(String department) {
        return employeeRepo.findByDepartment(department);
    }

    @Override
    public List<Employees> filterByRole(String role) {
        return employeeRepo.findByRole(role);
    }

    @Override
    public List<Employees> filterByLocation(String location) {
        return employeeRepo.findByLocation(location);
    }
}
```

EmployeeServiceInterface.java

```
package com.aditya.EmployeeManagementSystem.services;

import com.aditya.EmployeeManagementSystem.beans.Employees;

import java.util.List;

public interface EmployeeServiceInterface {

    public void addEmployee(Employees employees);
    public Employees getEmployeeById(int employeeId);

    public List<Employees> getAllEmployees();

    public void updateEmployee(Employees employees);

    public void deleteEmploye(int empId);

    List<Employees> filterByDepartment(String department);
    List<Employees> filterByRole(String role);
    List<Employees> filterByLocation(String location);
}
```

EmplyoeeRepo.java

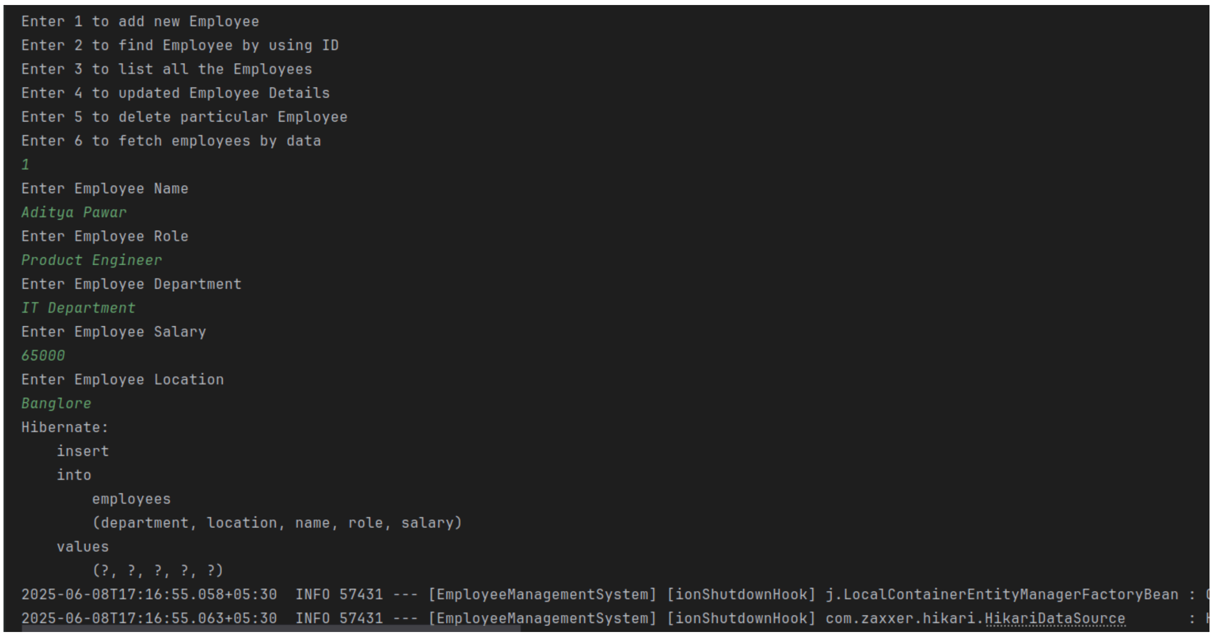
```
package com.aditya.EmployeeManagementSystem.repos;

import com.aditya.EmployeeManagementSystem.beans.Employees;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface EmployeeRepo extends JpaRepository<Employees,Integer> {
    List<Employees> findByDepartment(String department);
    List<Employees> findByRole(String role);
    List<Employees> findByLocation(String location);
}
```

Application.properties

```
spring.application.name=EmployeeManagementSystem
server.port=8082

spring.datasource.username=postgres
spring.datasource.password=root
spring.datasource.url=jdbc:postgresql://localhost:5432/dbspring

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDiale

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

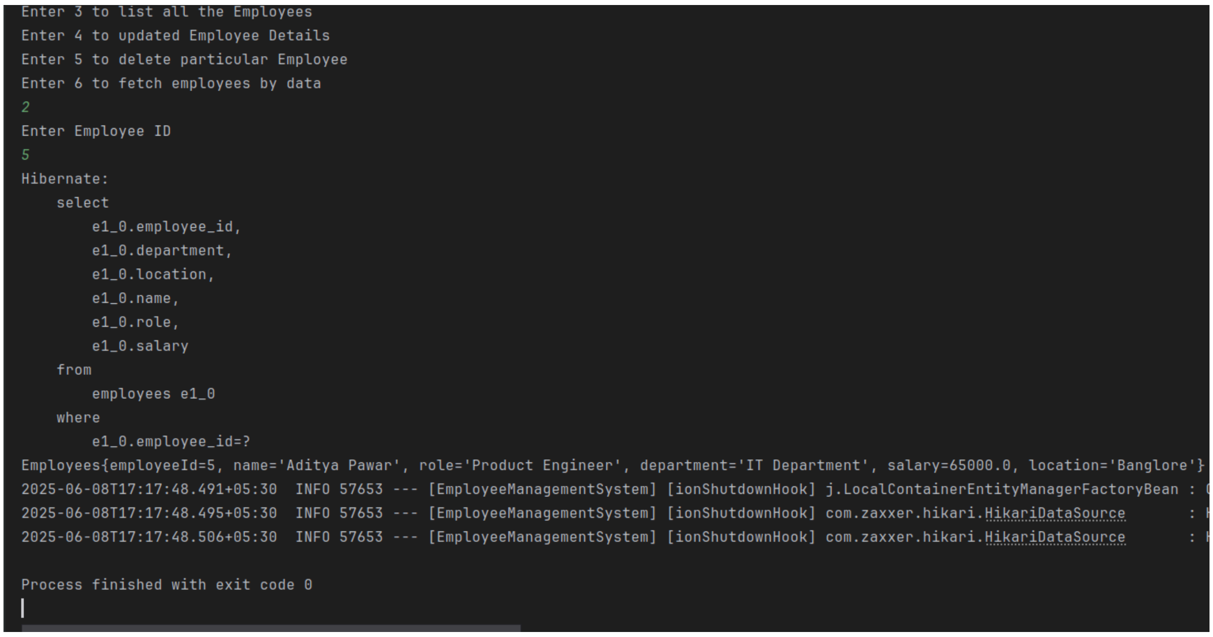
Output:

case 1 —>



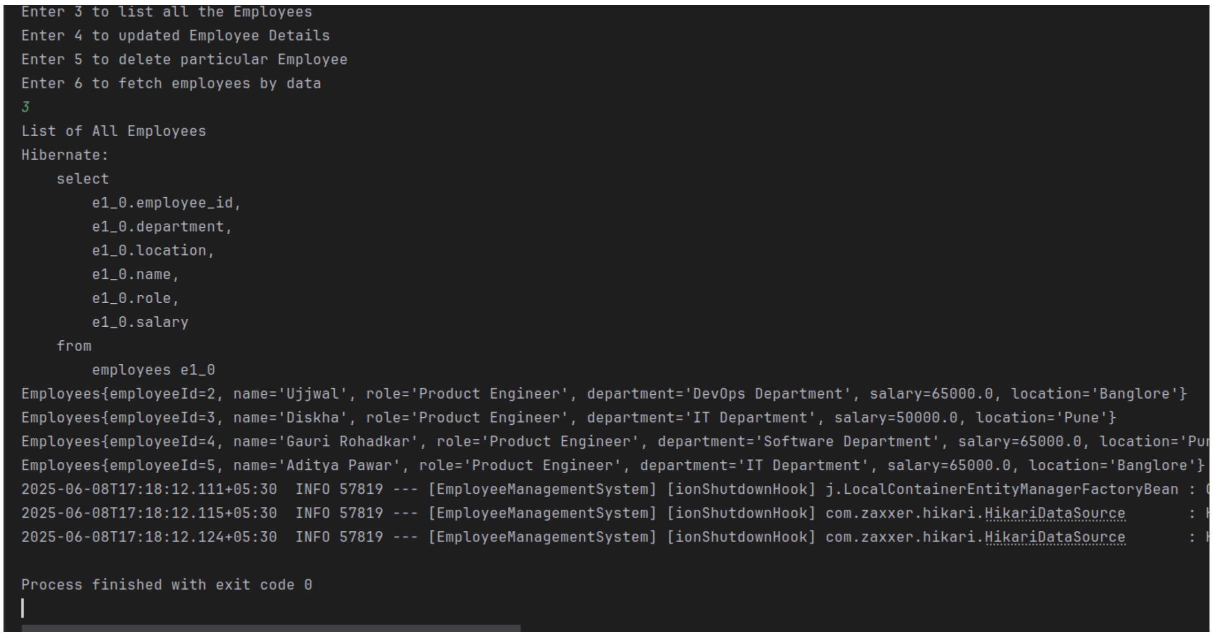
```
Enter 1 to add new Employee
Enter 2 to find Employee by using ID
Enter 3 to list all the Employees
Enter 4 to updated Employee Details
Enter 5 to delete particular Employee
Enter 6 to fetch employees by data
1
Enter Employee Name
Aditya Pawar
Enter Employee Role
Product Engineer
Enter Employee Department
IT Department
Enter Employee Salary
65000
Enter Employee Location
Banglore
Hibernate:
    insert
    into
        employees
        (department, location, name, role, salary)
    values
        (?, ?, ?, ?, ?)
2025-06-08T17:16:55.058+05:30  INFO 57431 --- [EmployeeManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : 
2025-06-08T17:16:55.063+05:30  INFO 57431 --- [EmployeeManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : 
```

case 2 —>



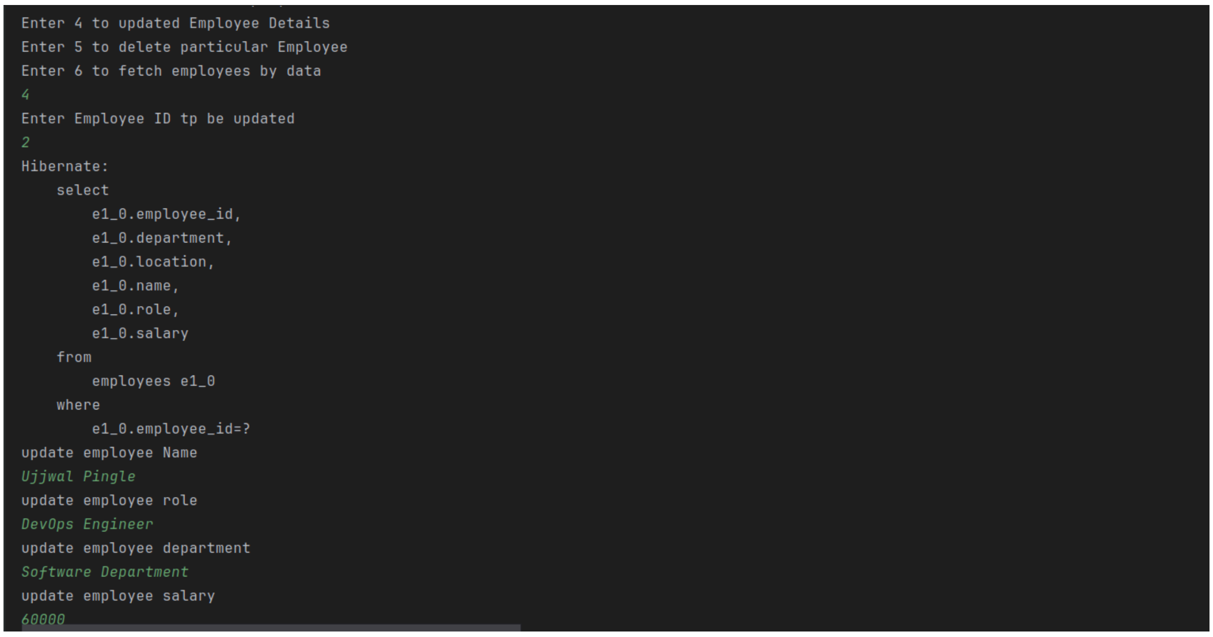
```
Enter 3 to list all the Employees
Enter 4 to updated Employee Details
Enter 5 to delete particular Employee
Enter 6 to fetch employees by data
2
Enter Employee ID
5
Hibernate:
    select
        e1_0.employee_id,
        e1_0.department,
        e1_0.location,
        e1_0.name,
        e1_0.role,
        e1_0.salary
    from
        employees e1_0
    where
        e1_0.employee_id=?
Employees{employeeId=5, name='Aditya Pawar', role='Product Engineer', department='IT Department', salary=65000.0, location='Banglore'}
2025-06-08T17:17:48.491+05:30  INFO 57653 --- [EmployeeManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : 
2025-06-08T17:17:48.495+05:30  INFO 57653 --- [EmployeeManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : 
2025-06-08T17:17:48.506+05:30  INFO 57653 --- [EmployeeManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : 

Process finished with exit code 0
```

case 3 —>

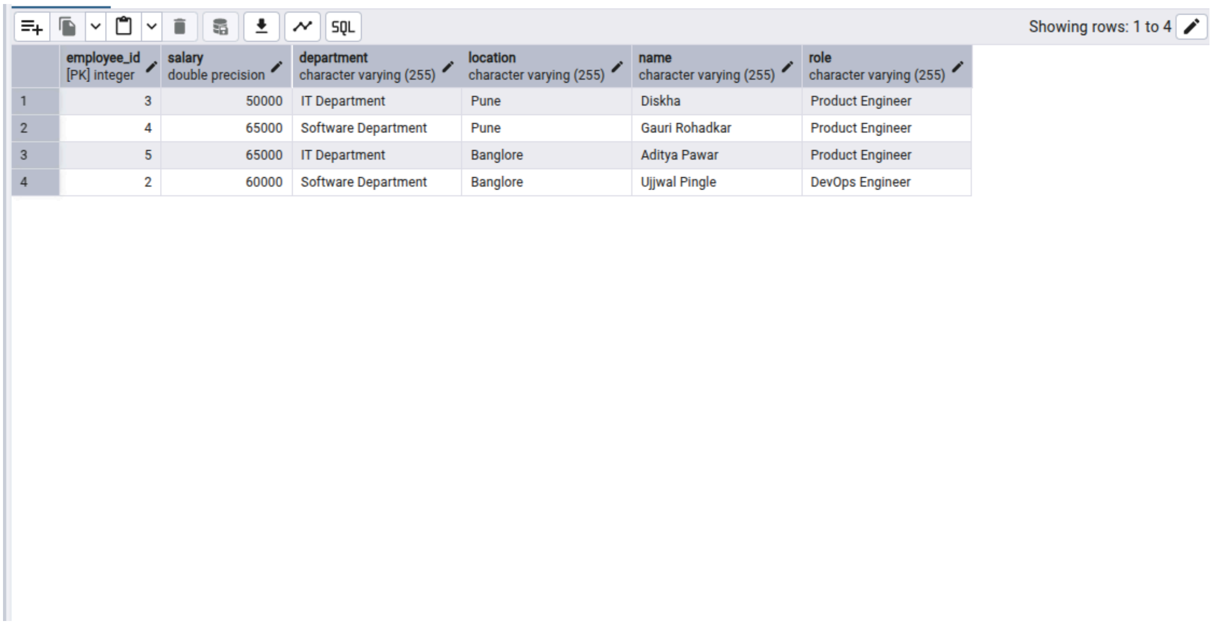


```
Enter 3 to list all the Employees
Enter 4 to updated Employee Details
Enter 5 to delete particular Employee
Enter 6 to fetch employees by data
3
List of All Employees
Hibernate:
    select
        e1_0.employee_id,
        e1_0.department,
        e1_0.location,
        e1_0.name,
        e1_0.role,
        e1_0.salary
    from
        employees e1_0
Employees{employeeId=2, name='Ujjwal', role='Product Engineer', department='DevOps Department', salary=65000.0, location='Banglore'}
Employees{employeeId=3, name='Diskha', role='Product Engineer', department='IT Department', salary=50000.0, location='Pune'}
Employees{employeeId=4, name='Gauri Rohadkar', role='Product Engineer', department='Software Department', salary=65000.0, location='Pu
Employees{employeeId=5, name='Aditya Pawar', role='Product Engineer', department='IT Department', salary=65000.0, location='Banglore'}
2025-06-08T17:18:12.111+05:30  INFO 57819 --- [EmployeeManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : 
2025-06-08T17:18:12.115+05:30  INFO 57819 --- [EmployeeManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : 
2025-06-08T17:18:12.124+05:30  INFO 57819 --- [EmployeeManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : 

Process finished with exit code 0
```

case 4 —>



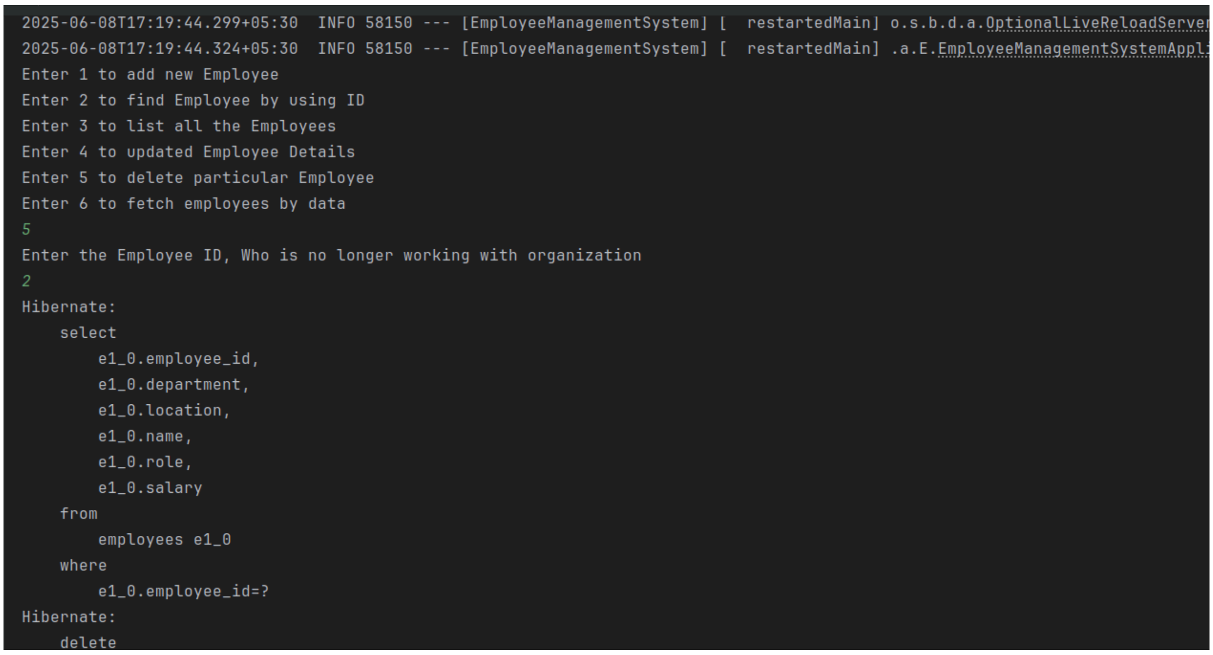
```
Enter 4 to updated Employee Details
Enter 5 to delete particular Employee
Enter 6 to fetch employees by data
4
Enter Employee ID tp be updated
2
Hibernate:
    select
        e1_0.employee_id,
        e1_0.department,
        e1_0.location,
        e1_0.name,
        e1_0.role,
        e1_0.salary
    from
        employees e1_0
    where
        e1_0.employee_id=?
update employee Name
Ujjwal Pingle
update employee role
DevOps Engineer
update employee department
Software Department
update employee salary
60000
```



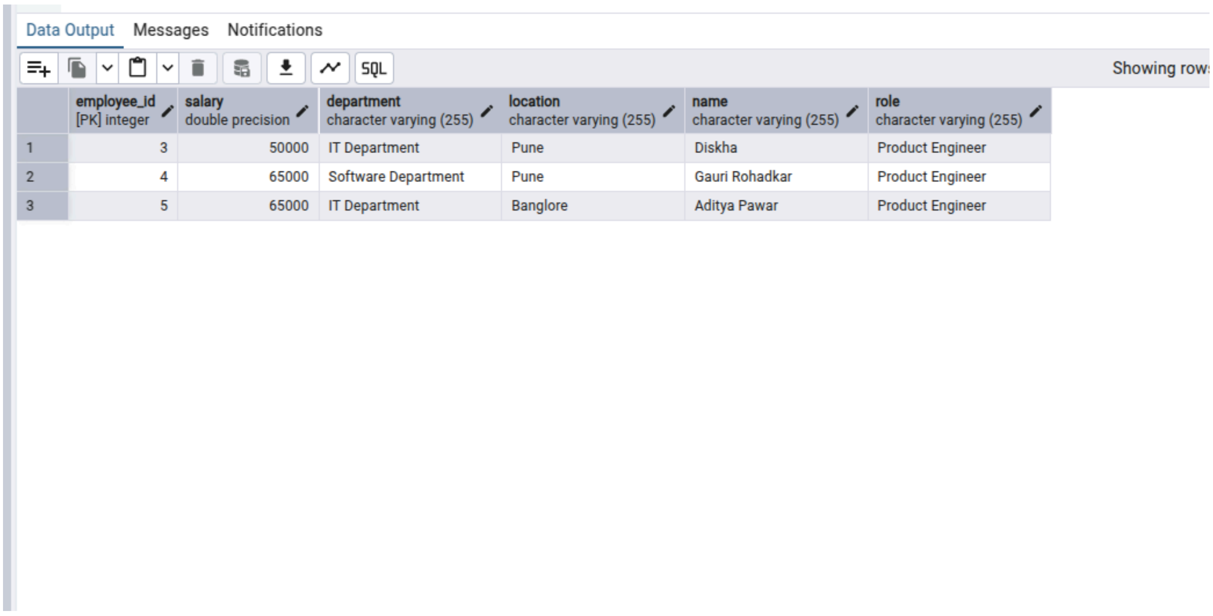
Showing rows: 1 to 4

	employee_id [PK] integer	salary double precision	department character varying (255)	location character varying (255)	name character varying (255)	role character varying (255)
1	3	50000	IT Department	Pune	Diskha	Product Engineer
2	4	65000	Software Department	Pune	Gauri Rohadkar	Product Engineer
3	5	65000	IT Department	Banglore	Aditya Pawar	Product Engineer
4	2	60000	Software Department	Banglore	Ujjwal Pingle	DevOps Engineer

case 5 —>



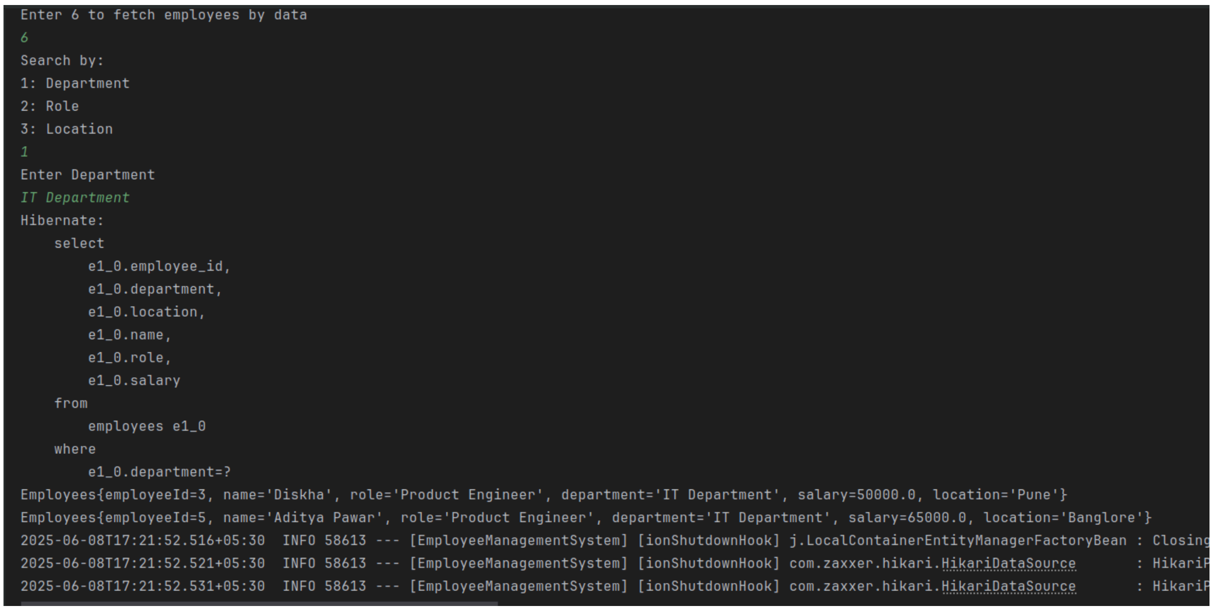
```
2025-06-08T17:19:44.299+05:30  INFO 58150 --- [EmployeeManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer
2025-06-08T17:19:44.324+05:30  INFO 58150 --- [EmployeeManagementSystem] [  restartedMain] .a.E.EmployeeManagementSystemAppli
Enter 1 to add new Employee
Enter 2 to find Employee by using ID
Enter 3 to list all the Employees
Enter 4 to updated Employee Details
Enter 5 to delete particular Employee
Enter 6 to fetch employees by data
5
Enter the Employee ID, Who is no longer working with organization
2
Hibernate:
    select
        e1_0.employee_id,
        e1_0.department,
        e1_0.location,
        e1_0.name,
        e1_0.role,
        e1_0.salary
    from
        employees e1_0
    where
        e1_0.employee_id=?
Hibernate:
    delete
```



	employee_id [PK] integer	salary double precision	department character varying (255)	location character varying (255)	name character varying (255)	role character varying (255)
1	3	50000	IT Department	Pune	Diskha	Product Engineer
2	4	65000	Software Department	Pune	Gauri Rohadkar	Product Engineer
3	5	65000	IT Department	Banglore	Aditya Pawar	Product Engineer

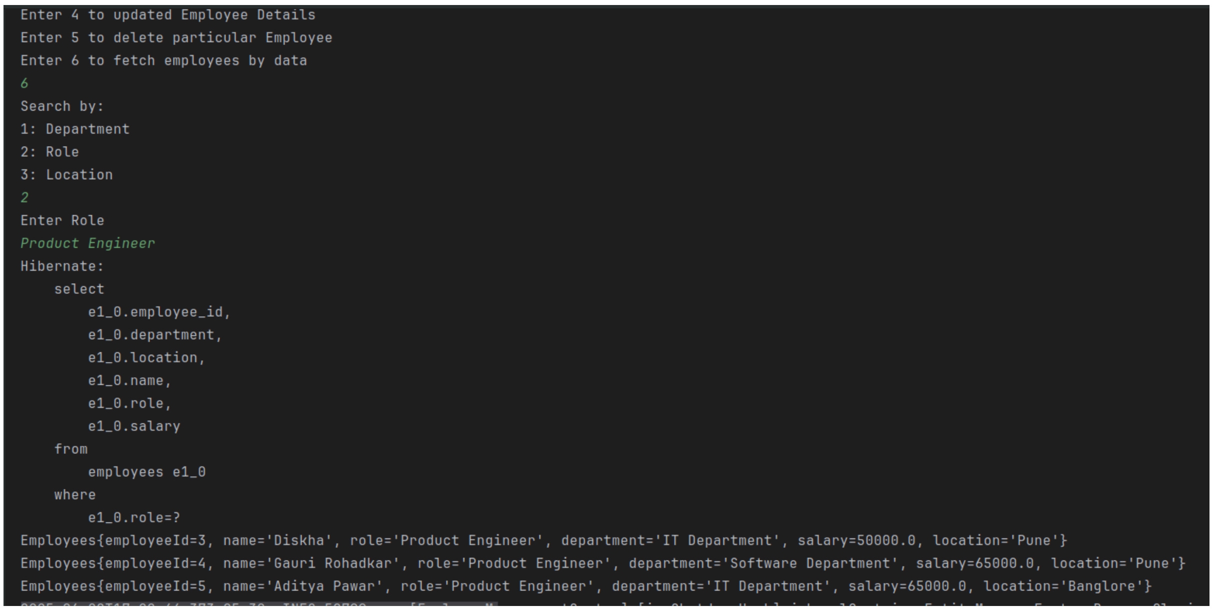
case 6 —>

1 by department



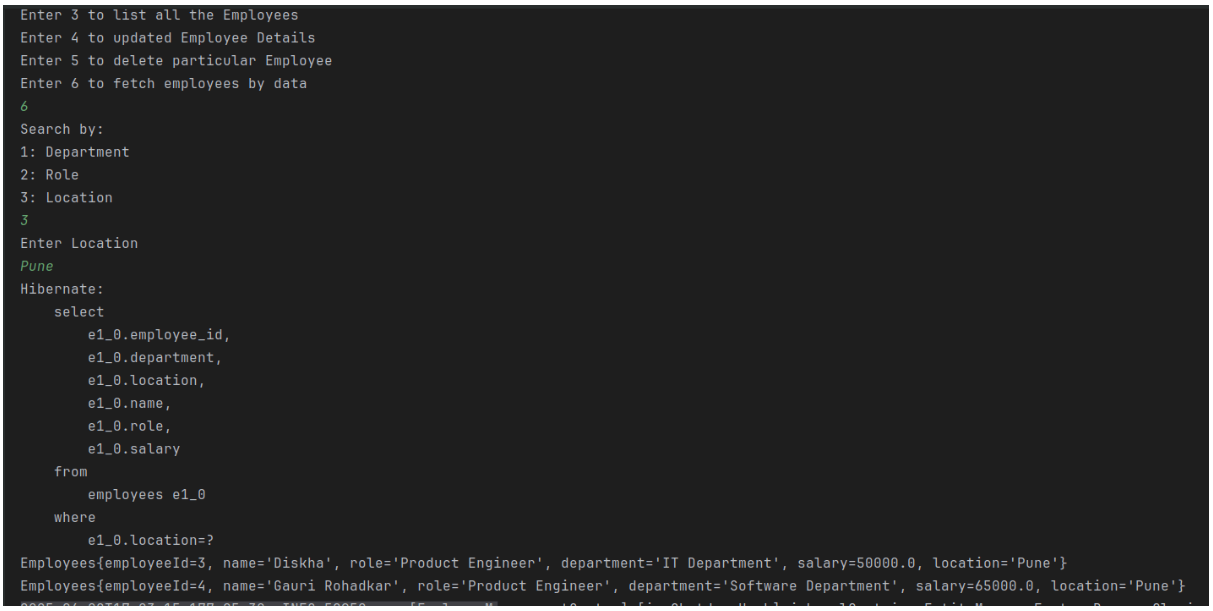
```
Enter 6 to fetch employees by data
6
Search by:
1: Department
2: Role
3: Location
1
Enter Department
IT Department
Hibernate:
    select
        e1_0.employee_id,
        e1_0.department,
        e1_0.location,
        e1_0.name,
        e1_0.role,
        e1_0.salary
    from
        employees e1_0
    where
        e1_0.department=?
Employees{employeeId=3, name='Diskha', role='Product Engineer', department='IT Department', salary=50000.0, location='Pune'}
Employees{employeeId=5, name='Aditya Pawar', role='Product Engineer', department='IT Department', salary=65000.0, location='Banglore'}
2025-06-08T17:21:52.516+05:30  INFO 58613 --- [EmployeeManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing
2025-06-08T17:21:52.521+05:30  INFO 58613 --- [EmployeeManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariP
2025-06-08T17:21:52.531+05:30  INFO 58613 --- [EmployeeManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariP
```

2 by role



```
Enter 4 to updated Employee Details
Enter 5 to delete particular Employee
Enter 6 to fetch employees by data
6
Search by:
1: Department
2: Role
3: Location
2
Enter Role
Product Engineer
Hibernate:
    select
        e1_0.employee_id,
        e1_0.department,
        e1_0.location,
        e1_0.name,
        e1_0.role,
        e1_0.salary
    from
        employees e1_0
    where
        e1_0.role=?
Employees{employeeId=3, name='Diskha', role='Product Engineer', department='IT Department', salary=50000.0, location='Pune'}
Employees{employeeId=4, name='Gauri Rohadkar', role='Product Engineer', department='Software Department', salary=65000.0, location='Pune'}
Employees{employeeId=5, name='Aditya Pawar', role='Product Engineer', department='IT Department', salary=65000.0, location='Banglore'}
```

3 by Location



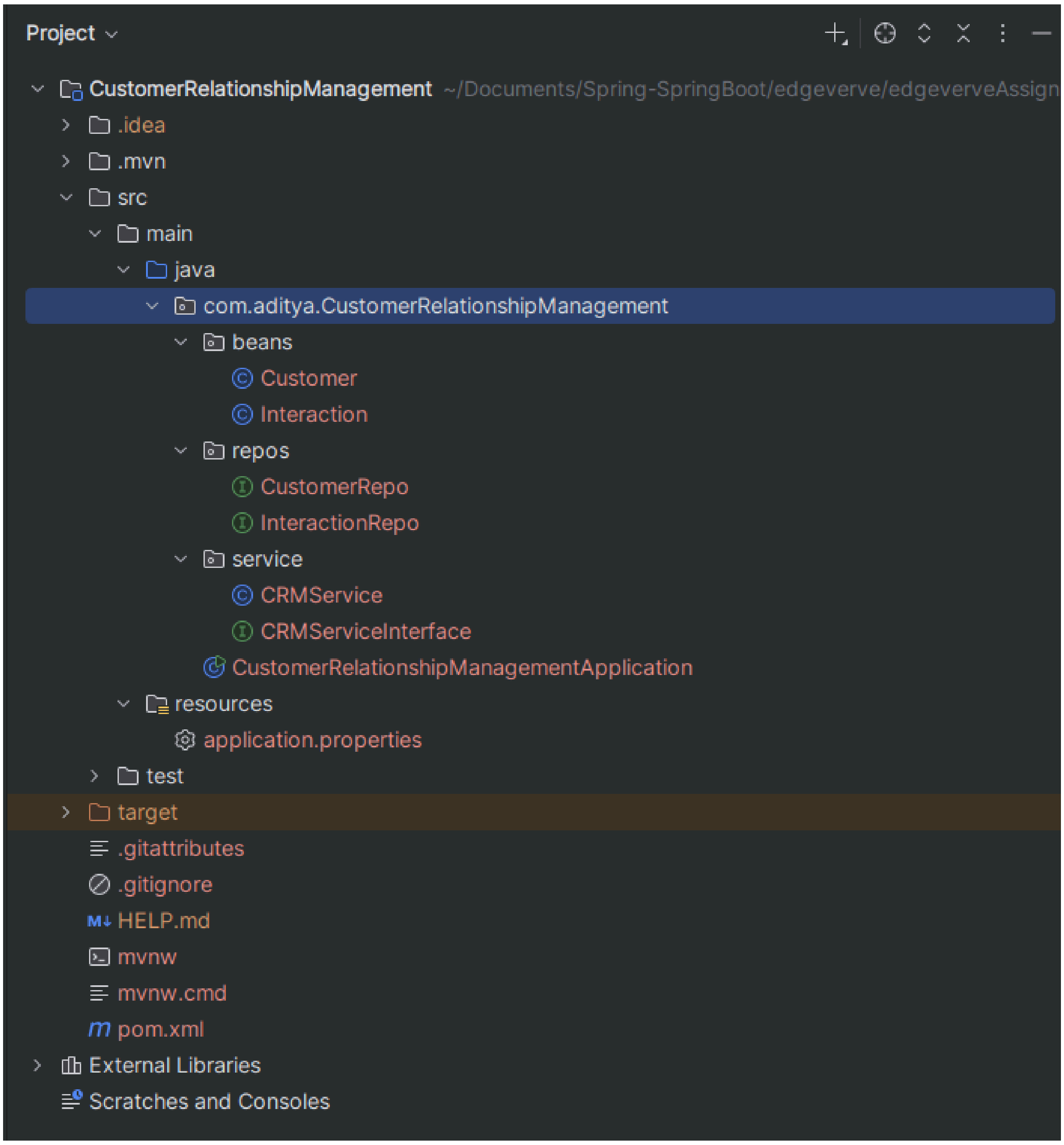
```
Enter 3 to list all the Employees
Enter 4 to updated Employee Details
Enter 5 to delete particular Employee
Enter 6 to fetch employees by data
6
Search by:
1: Department
2: Role
3: Location
3
Enter Location
Pune
Hibernate:
    select
        e1_0.employee_id,
        e1_0.department,
        e1_0.location,
        e1_0.name,
        e1_0.role,
        e1_0.salary
    from
        employees e1_0
    where
        e1_0.location=?
Employees{employeeId=3, name='Diskha', role='Product Engineer', department='IT Department', salary=50000.0, location='Pune'}
Employees{employeeId=4, name='Gauri Rohadkar', role='Product Engineer', department='Software Department', salary=65000.0, location='Pune'}
```

Que. 3 Customer Relationship Management (CRM) System

Problem Statement: Develop a CRM system to manage customer interactions and data. The system should enable:

- Create operations to add new customer profiles with contact information and interaction history.
- Read functionalities to view customer details and interaction logs.
- Update capabilities to modify customer information or update interaction records.
- Delete operations to remove inactive or unresponsive customers.
- Fetch feedback for a specific product, within a specific date range, or with a particular rating.

Folder Structure:



CustomerRelationshipManagementApplication.java

```
package com.aditya.CustomerRelationshipManagement;

import com.aditya.CustomerRelationshipManagement.beans.Customer;
import com.aditya.CustomerRelationshipManagement.beans.Interaction;
import com.aditya.CustomerRelationshipManagement.service.CRMService;
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.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Scanner;

@SpringBootApplication
public class CustomerRelationshipManagementApplication implements
CommandLineRunner {

    @Autowired
    CRMService crmService;

    public static void main(String[] args) {
        SpringApplication.run(CustomerRelationshipManagementApplication.clas
    }

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

        System.out.println("Enter 1 to Add new Customer");
        System.out.println("Enter 2 to View Customer");
        System.out.println("Enter 3 to Update Customer");
        System.out.println("Enter 4 to Delete Customer");
        System.out.println("Enter 5 to Add Interaction");
        System.out.println("Enter 6 to view Feedback by Product");
        System.out.println("Enter 7 to view Feedback by Date Range");
        System.out.println("Enter 8 to view Feedback by Rating");

        int operation = scanner.nextInt();
        scanner.nextLine();

        switch (operation)
        {
            case 1:
                System.out.println("Enter the Customer Details:");
                Customer customer = new Customer();

                System.out.println("Enter Customer Name");
                String name = scanner.nextLine();
                System.out.println("Enter customer email");
                String email = scanner.nextLine();
                System.out.println("Enter Customer Contact Number");
                String phone = scanner.nextLine();
                System.out.println("enter customer address");
                String address = scanner.nextLine();

                customer.setName(name);
                customer.setEmail(email);
                customer.setPhone(phone);
                customer.setAddress(address);
                crmService.addCustomer(customer);
                break;

            case 2:
                System.out.println("View customer:");
                System.out.println("1. Using Customer ID...Enter Id Below");
                System.out.println("2. List All Customers");

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

                switch (option) {
                    case 1:
                    System.out.println("Enter Customer ID");
                    int customerId = scanner.nextInt();
                    scanner.nextLine();
                    try {
                        Customer customer1 = crmService.getCustomerById(customerI
                        System.out.println(customer1.toString());
                    } catch (Exception e) {
                        System.err.println("Id not valid" + e.getMessage());
                    }
                    break;
                    case 2:
                        try
                        {
                            List<Customer> customerList = crmService.getAllCustomer()
                            customerList.forEach(customer1 → System.out.println(custon
                        } catch (Exception e) {
                            System.out.println("error displaying records"+e.getMessage(
                        }
                        break;

                    default:
                        System.out.println("choose correct option!!!");
                }
                break;

            case 3:
                System.out.println("Enter Customer Id to be Updated");
                int id = scanner.nextInt();
                scanner.nextLine();

                try
                {
                    Customer existingCustomer = crmService.getCustomerById(id);

                    if (existingCustomer == null)
                    {
                        System.out.println("Employee with ID"+id+" Not found");
                        break;
                    }

                    System.out.println("Update Customer Name");
                    String newName = scanner.nextLine();
                    System.out.println("Update Customer Email");
                    String newEmail = scanner.nextLine();
                    System.out.println("Update Customer phone");
                    String newPhone = scanner.nextLine();
                    System.out.println("Update Customer Address");
                    String newAddr = scanner.nextLine();

                    existingCustomer.setName(newName);
                    existingCustomer.setEmail(newEmail);
                    existingCustomer.setPhone(newPhone);
                    existingCustomer.setAddress(newAddr);
                    crmService.updateCustomer(existingCustomer);
                    System.out.println("Customer Updated Successfully...");
                } catch (Exception e) {
                    System.err.println("Id not valid"+e.getMessage());
                }
                break;

            case 4:
                System.out.println("Enter customer id");
                id = scanner.nextInt();
                scanner.nextLine();
                crmService.deleteCustomer(id);
                System.out.println("Customer "+id+" deleted Successfully...");
                break;

            case 5:
                System.out.print("Enter Customer ID: ");
                int customerId = scanner.nextInt();
                scanner.nextLine();

                System.out.print("Enter Product: ");
                String product = scanner.nextLine();

                System.out.print("Enter Rating (1-5): ");
                int rating = scanner.nextInt();
                scanner.nextLine();

                System.out.print("Enter Interaction Notes: ");
                String notes = scanner.nextLine();

                Interaction interaction = new Interaction();
                interaction.setProduct(product);
                interaction.setRating(rating);
                interaction.setNotes(notes);

                try {
                    crmService.addInteraction(customerId, interaction);
                    System.out.println(" Interaction added successfully.");
                } catch (Exception e) {
                    System.out.println(" Failed to add interaction: " + e.getMessage())
                }
                break;

            case 6:
                System.out.println("Enter Product Name:");
                String prod = scanner.nextLine();
                List<Interaction> productFeedback = crmService.getFeedbackByPro
                productFeedback.forEach(prod1 → System.out.println(prod1));
                break;

            case 7:
                DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-

                System.out.println("Enter Start Date (yyyy-MM-ddd): ");
                String startStr = scanner.nextLine();
                LocalDate startDate = LocalDate.parse(startStr, formatter);

                System.out.println("Enter End Date (yyyy-MM-dd): ");
                String endStr = scanner.nextLine();
                LocalDate endDate = LocalDate.parse(endStr, formatter);

                List<Interaction> dateFeedback = crmService.getFeedbackByDateR
                startDate, endDate);
                dateFeedback.forEach(date → System.out.println(date));
                break;

            case 8:
                System.out.print("Enter Rating (1 to 5): ");
                int rating1 = scanner.nextInt();
                scanner.nextLine();
```

```
                List<Interaction> ratingFeedback = crmService.getFeedbackByRatin
                rating1);
                ratingFeedback.forEach(rate → System.out.println(rate));
                break;

            default:
                System.out.println("Invalid Input!!!");
        }
    }
}
```

CRMServiceInterface.java

```
package com.aditya.CustomerRelationshipManagement.service;


import com.aditya.CustomerRelationshipManagement.beans.Customer;
import com.aditya.CustomerRelationshipManagement.beans.Interaction;

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

public interface CRMServiceInterface {

    public Customer addCustomer(Customer customer);

    public Customer getCustomerById(int id);

    public List<Customer> getAllCustomer();

    public void deleteCustomer(int id);

    public Customer updateCustomer(Customer customer);

    public Interaction addInteraction(int customerId,Interaction interaction);

    public List<Interaction> getFeedbackByProduct(String product);

    public List<Interaction> getFeedbackByDateRange(LocalDate from, LocalDa

    public List<Interaction> getFeedbackByRating(int rating);
}
```

CRMService.java

```
package com.aditya.CustomerRelationshipManagement.service;

import com.aditya.CustomerRelationshipManagement.beans.Customer;
import com.aditya.CustomerRelationshipManagement.beans.Interaction;
import com.aditya.CustomerRelationshipManagement.repos.CustomerRepo;
import com.aditya.CustomerRelationshipManagement.repos.InteractionRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

@Service
public class CRMService implements CRMServiceInterface {

    @Autowired
    private CustomerRepo customerRepo;

    @Autowired
    private InteractionRepo interactionRepo;

    // Customer Services

    @Override
    public Customer addCustomer(Customer customer) {
        return customerRepo.save(customer);
    }
```

```
@Override
public Customer getCustomerById(int id) {
    return customerRepo.findById(id).orElse(null);
}

@Override
public List<Customer> getAllCustomer() {
    return customerRepo.findAll();
}

@Override
public void deleteCustomer(int id) {
    customerRepo.deleteById(id);
}

@Override
public Customer updateCustomer(Customer customer) {
    return customerRepo.save(customer);
}

// Interaction Services

@Override
public Interaction addInteraction(int customerId, Interaction interaction) {
    Optional<Customer> customerOpt = customerRepo.findById(customerId
    if (customerOpt.isPresent()) {
        interaction.setCustomer(customerOpt.get());
        interaction.setDate(LocalDate.now());
        return interactionRepo.save(interaction);
    } else {
        throw new IllegalArgumentException("Customer with ID " +
        customerId + " not found...");
    }
}

@Override
public List<Interaction> getFeedbackByProduct(String product) {
        return interactionRepo.findByProduct(product);
    }

    @Override
    public List<Interaction> getFeedbackByDateRange(LocalDate start,
    LocalDate end) {
        return interactionRepo.findByDateBetween(start, end);
    }

    @Override
    public List<Interaction> getFeedbackByRating(int rating) {
        return interactionRepo.findByRating(rating);
    }

}
```

Customer.java

```
package com.aditya.CustomerRelationshipManagement.beans;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

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

@NoArgsConstructor
@Entity
@Table(name = "customer")
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private @Getter @Setter int customerId;
    @NotBlank
    private @Getter @Setter String name;

    private @Getter @Setter String email;
    private @Getter @Setter String phone;
    private @Getter @Setter String address;

    @OneToMany(mappedBy = "customer", cascade = CascadeType.ALL)
    private List<Interaction> interactions = new ArrayList<>();

    public Customer(int customerId, String name, String email, String address,
    String phone) {
        this.customerId = customerId;
        this.name = name;
        this.email = email;
        this.address = address;
        this.phone = phone;
    }

    @Override
    public String toString() {
        return "Customer{" +
                "customerId=" + customerId +
                ", name='" + name + '\'' +
                ", email='" + email + '\'' +
                ", phone='" + phone + '\'' +
                ", address='" + address + '\'' +
                '}';
    }
}
```

Interaction.java

```
package com.aditya.CustomerRelationshipManagement.beans;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.time.LocalDate;

@NoArgsConstructor
@Entity
@Table(name = "interactions")
public class Interaction {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private @Getter @Setter int interactionId;

    @ManyToOne
    @JoinColumn(name = "customer_id", nullable = false)
    private @Getter @Setter Customer customer;

    private @Getter @Setter LocalDate date;
    private @Getter @Setter String notes;

    private @Getter @Setter String product;
    private @Getter @Setter int rating;

    public Interaction(int interactionId, Customer customer, LocalDate date,
    String notes, String product, int rating) {
        this.interactionId = interactionId;
        this.customer = customer;
        this.date = date;
        this.notes = notes;
        this.product = product;
        this.rating = rating;
    }

    @Override
    public String toString() {
        return "Interaction{" +
            "interactionId=" + interactionId +
            ", customer=" + customer +
            ", date=" + date +
            ", notes='" + notes + '\'' +
            ", ratedProducts='" + product + '\'' +
            ", rating=" + rating +
            '}';
    }
}
```

CustomerRepo.java

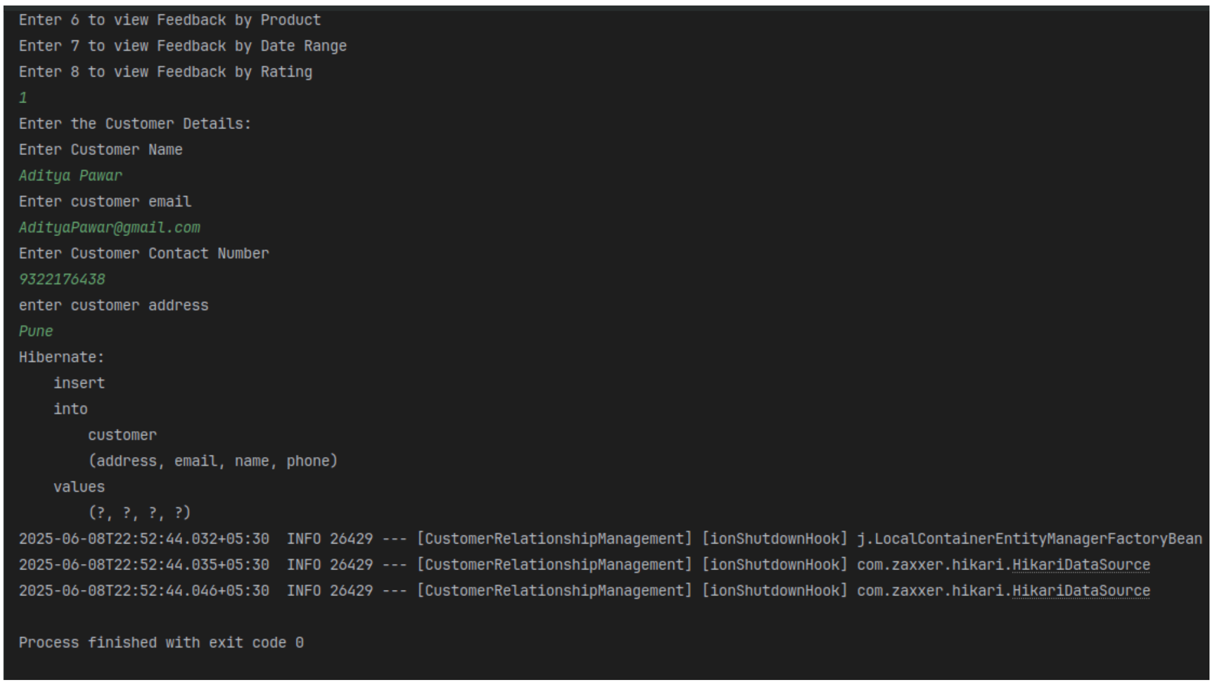
```
package com.aditya.CustomerRelationshipManagement.repos;

import com.aditya.CustomerRelationshipManagement.beans.Customer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CustomerRepo extends JpaRepository<Customer, Integer> {
}
```

InterfaceRepo.java

```
package com.aditya.CustomerRelationshipManagement.repos;

import com.aditya.CustomerRelationshipManagement.beans.Interaction;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

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

@Repository
public interface InteractionRepo extends JpaRepository<Interaction,Integer> 

    List<Interaction> findByProduct(String product);
    List<Interaction> findByDateBetween(LocalDate start, LocalDate end);

    List<Interaction> findByRating(int rating);

}
```

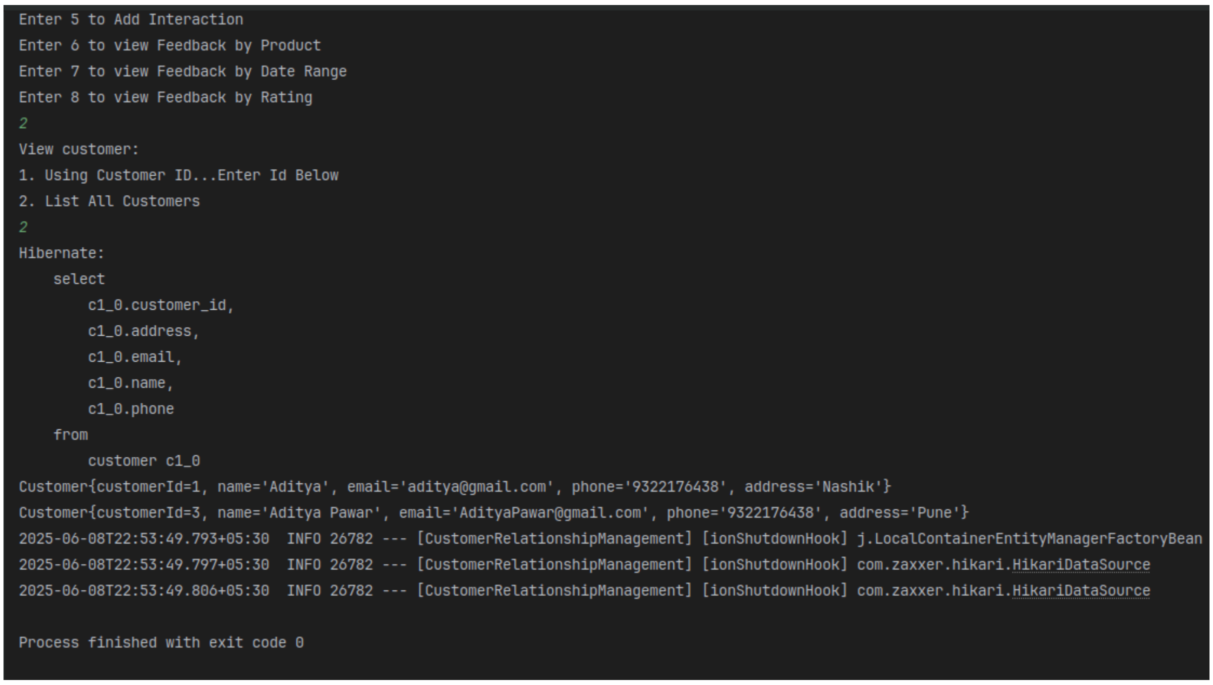
Output:

case 1 →



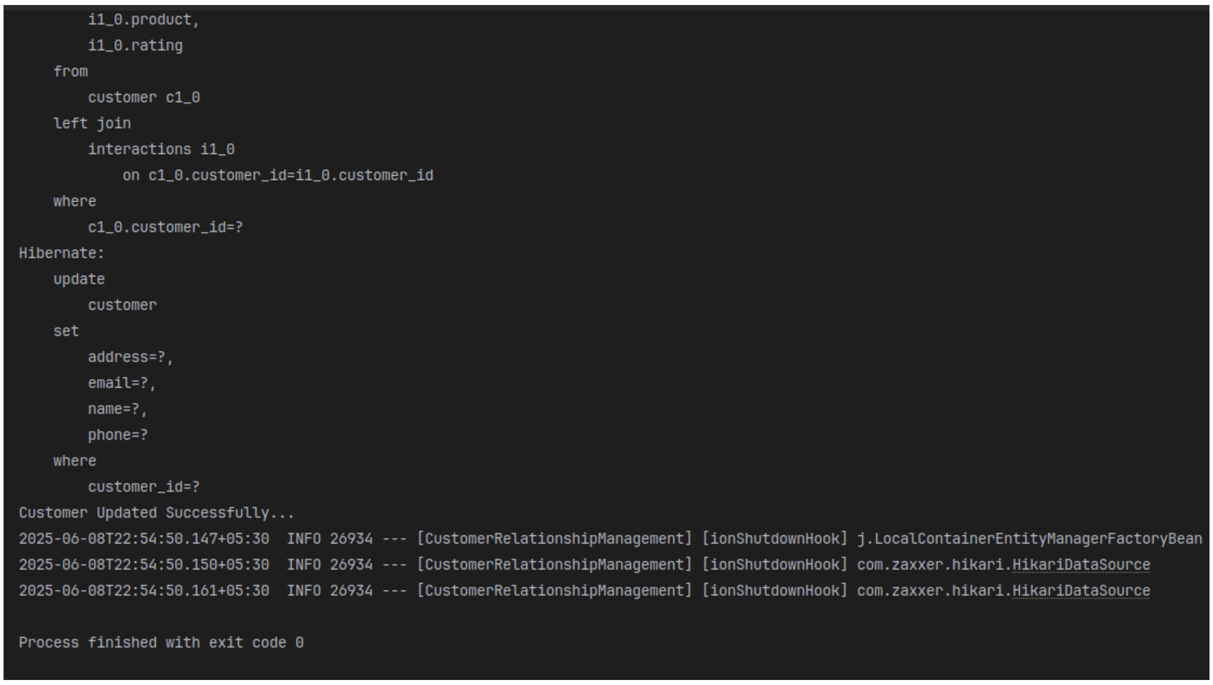
```
Enter 6 to view Feedback by Product
Enter 7 to view Feedback by Date Range
Enter 8 to view Feedback by Rating
1
Enter the Customer Details:
Enter Customer Name
Aditya Pawar
Enter customer email
AdityaPawar@gmail.com
Enter Customer Contact Number
9322176438
enter customer address
Pune
Hibernate:
    insert
    into
        customer
        (address, email, name, phone)
    values
        (?, ?, ?, ?)
2025-06-08T22:52:44.032+05:30  INFO 26429 --- [CustomerRelationshipManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2025-06-08T22:52:44.035+05:30  INFO 26429 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2025-06-08T22:52:44.046+05:30  INFO 26429 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0
```

case 2 →



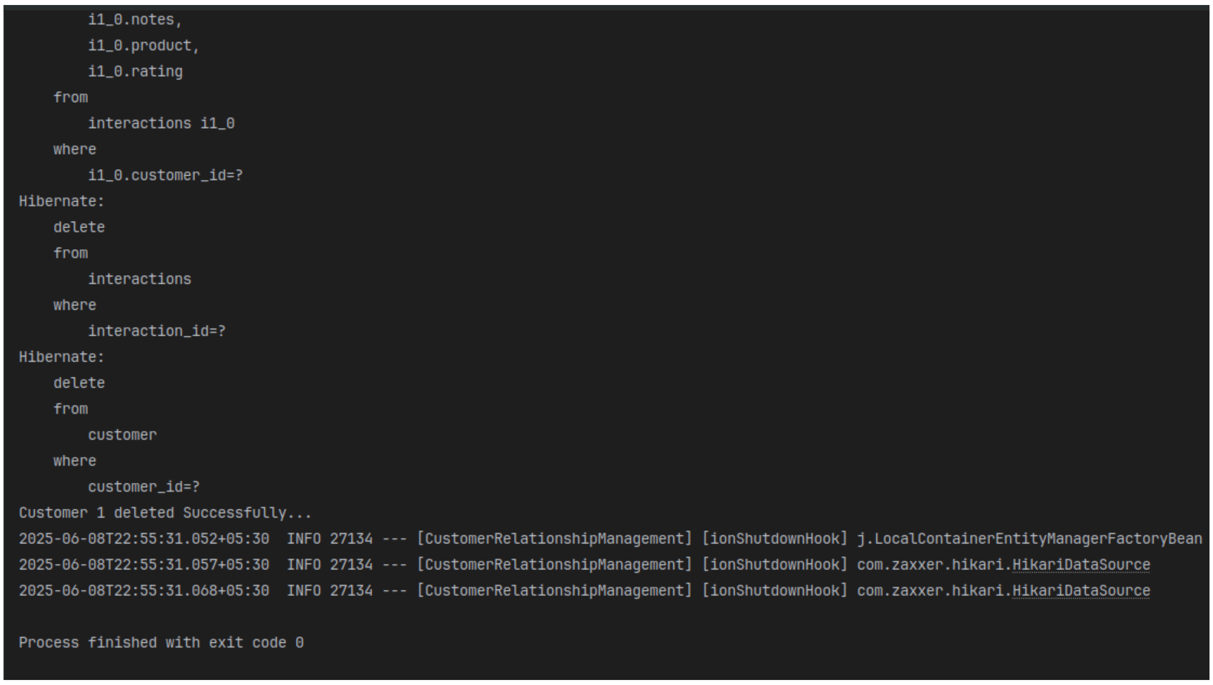
```
Enter 5 to Add Interaction
Enter 6 to view Feedback by Product
Enter 7 to view Feedback by Date Range
Enter 8 to view Feedback by Rating
2
View customer:
1. Using Customer ID...Enter Id Below
2. List All Customers
2
Hibernate:
    select
        c1_0.customer_id,
        c1_0.address,
        c1_0.email,
        c1_0.name,
        c1_0.phone
    from
        customer c1_0
Customer{customerId=1, name='Aditya', email='aditya@gmail.com', phone='9322176438', address='Nashik'}
Customer{customerId=3, name='Aditya Pawar', email='AdityaPawar@gmail.com', phone='9322176438', address='Pune'}
2025-06-08T22:53:49.793+05:30  INFO 26782 --- [CustomerRelationshipManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2025-06-08T22:53:49.797+05:30  INFO 26782 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2025-06-08T22:53:49.806+05:30  INFO 26782 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0
```

case 3 →

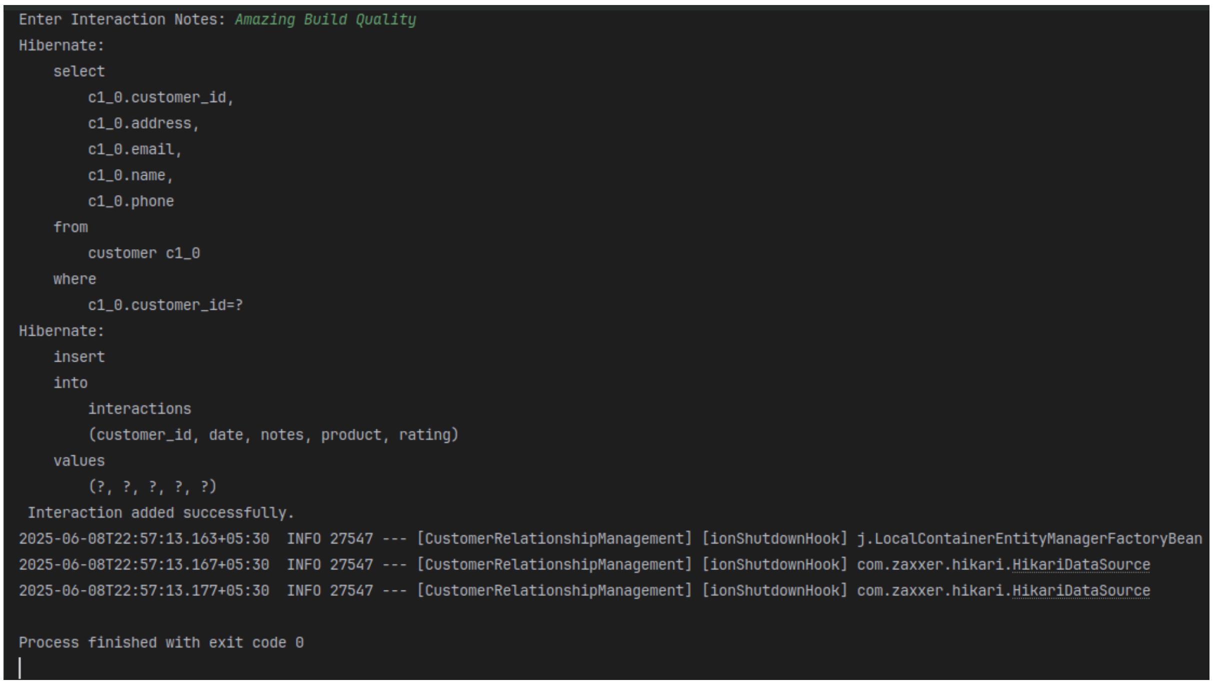


```
        i1_0.product,
        i1_0.rating
    from
        customer c1_0
    left join
        interactions i1_0
            on c1_0.customer_id=i1_0.customer_id
    where
        c1_0.customer_id=?
Hibernate:
    update
        customer
    set
        address=?,
        email=?,
        name=?,
        phone=?
    where
        customer_id=?
Customer Updated Successfully...
2025-06-08T22:54:50.147+05:30  INFO 26934 --- [CustomerRelationshipManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2025-06-08T22:54:50.150+05:30  INFO 26934 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2025-06-08T22:54:50.161+05:30  INFO 26934 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0
```

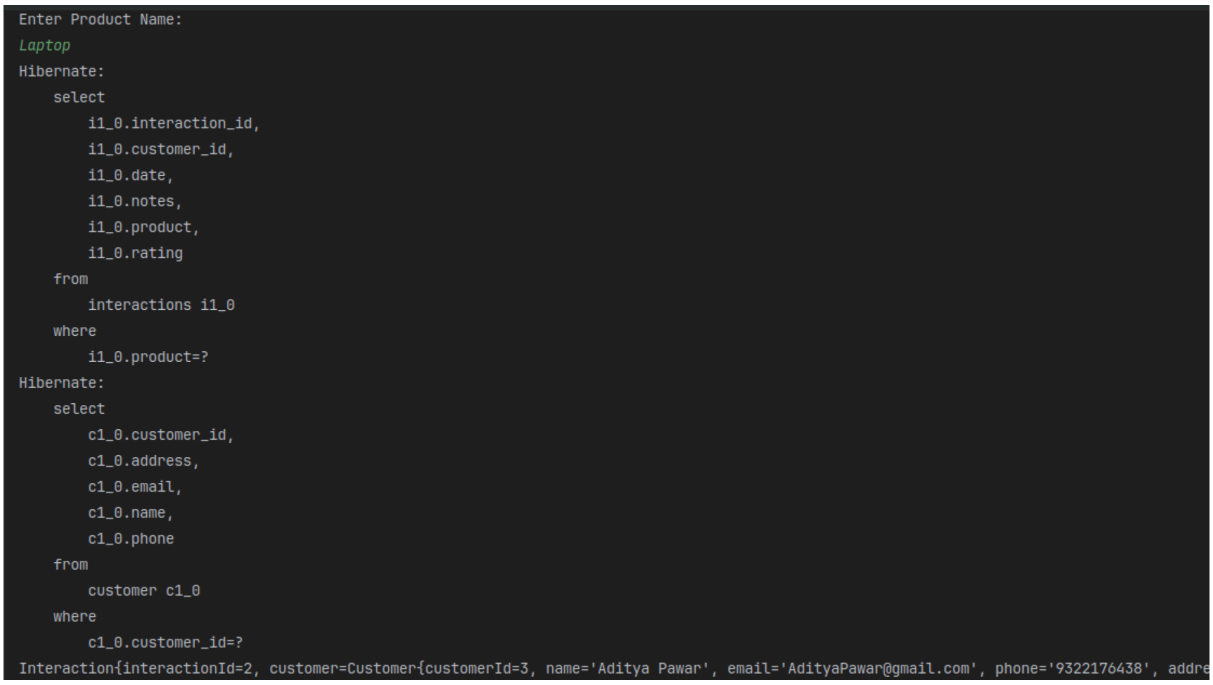
case 4 →


```
        i1_0.notes,
        i1_0.product,
        i1_0.rating
    from
        interactions i1_0
    where
        i1_0.customer_id=?
Hibernate:
    delete
    from
        interactions
    where
        interaction_id=?
Hibernate:
    delete
    from
        customer
    where
        customer_id=?
Customer 1 deleted Successfully...
2025-06-08T22:55:31.052+05:30  INFO 27134 --- [CustomerRelationshipManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2025-06-08T22:55:31.057+05:30  INFO 27134 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2025-06-08T22:55:31.068+05:30  INFO 27134 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0
```

case 5 →

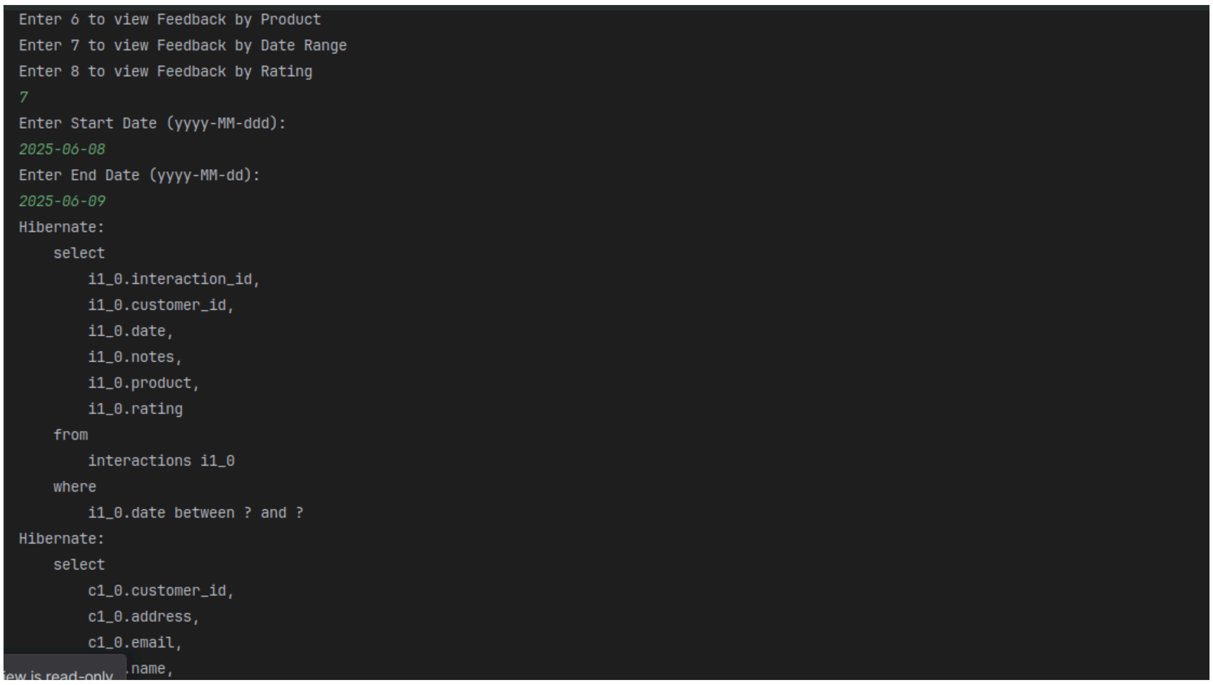

```
Enter Interaction Notes: Amazing Build Quality
Hibernate:
    select
        c1_0.customer_id,
        c1_0.address,
        c1_0.email,
        c1_0.name,
        c1_0.phone
    from
        customer c1_0
    where
        c1_0.customer_id=?
Hibernate:
    insert
    into
        interactions
        (customer_id, date, notes, product, rating)
    values
        (?, ?, ?, ?, ?)
 Interaction added successfully.
2025-06-08T22:57:13.163+05:30  INFO 27547 --- [CustomerRelationshipManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2025-06-08T22:57:13.167+05:30  INFO 27547 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2025-06-08T22:57:13.177+05:30  INFO 27547 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0
```

case 6 →



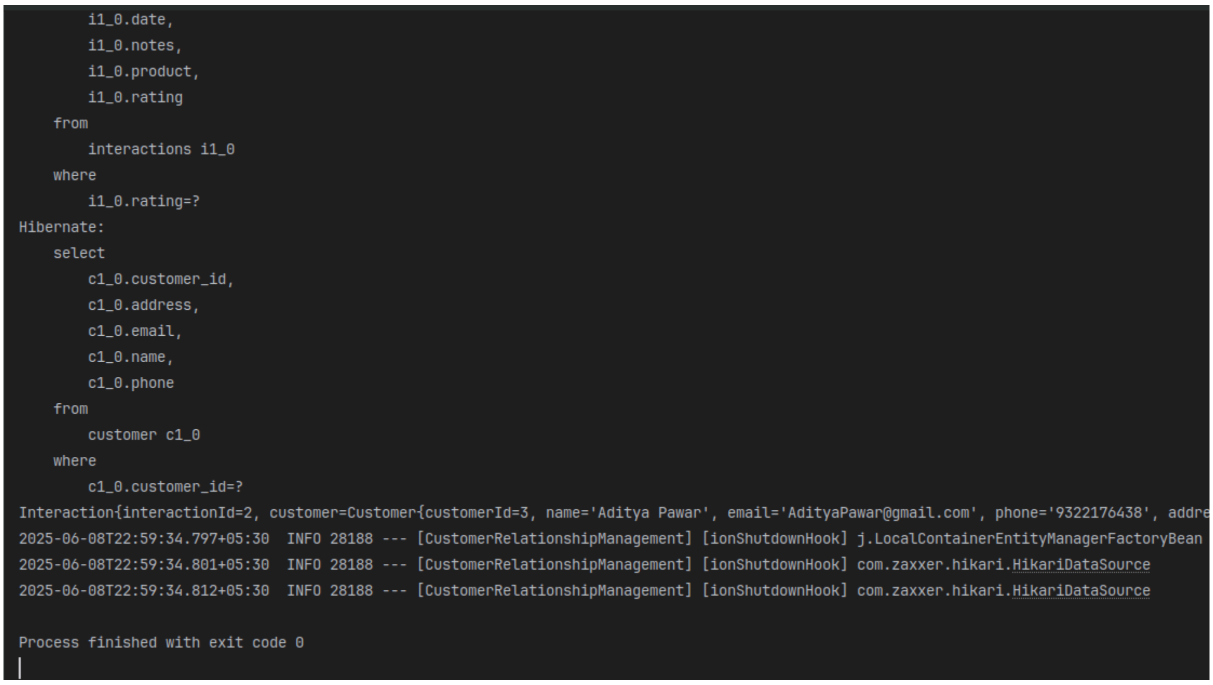
```
Enter Product Name:
Laptop
Hibernate:
    select
        i1_0.interaction_id,
        i1_0.customer_id,
        i1_0.date,
        i1_0.notes,
        i1_0.product,
        i1_0.rating
    from
        interactions i1_0
    where
        i1_0.product=?
Hibernate:
    select
        c1_0.customer_id,
        c1_0.address,
        c1_0.email,
        c1_0.name,
        c1_0.phone
    from
        customer c1_0
    where
        c1_0.customer_id=?
Interaction{interactionId=2, customer=Customer{customerId=3, name='Aditya Pawar', email='AdityaPawar@gmail.com', phone='9322176438', addre
```

case 7 →

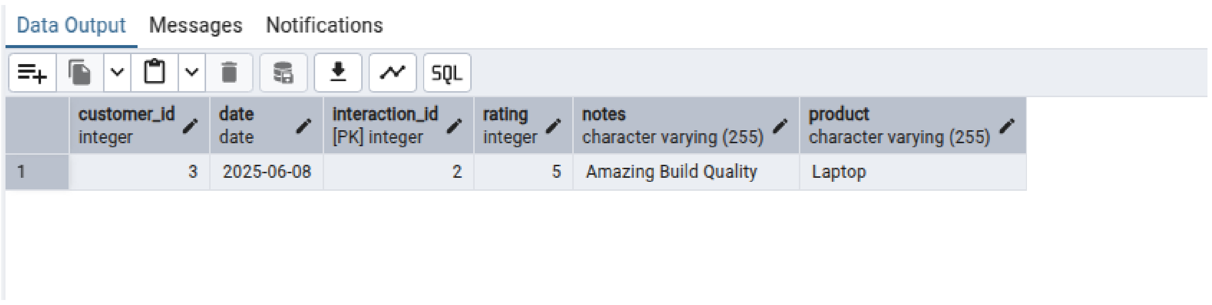


```
Enter 6 to view Feedback by Product
Enter 7 to view Feedback by Date Range
Enter 8 to view Feedback by Rating
7
Enter Start Date (yyyy-MM-ddd):
2025-06-08
Enter End Date (yyyy-MM-dd):
2025-06-09
Hibernate:
    select
        i1_0.interaction_id,
        i1_0.customer_id,
        i1_0.date,
        i1_0.notes,
        i1_0.product,
        i1_0.rating
    from
        interactions i1_0
    where
        i1_0.date between ? and ?
Hibernate:
    select
        c1_0.customer_id,
        c1_0.address,
        c1_0.email,
iew is read-only .name,
```

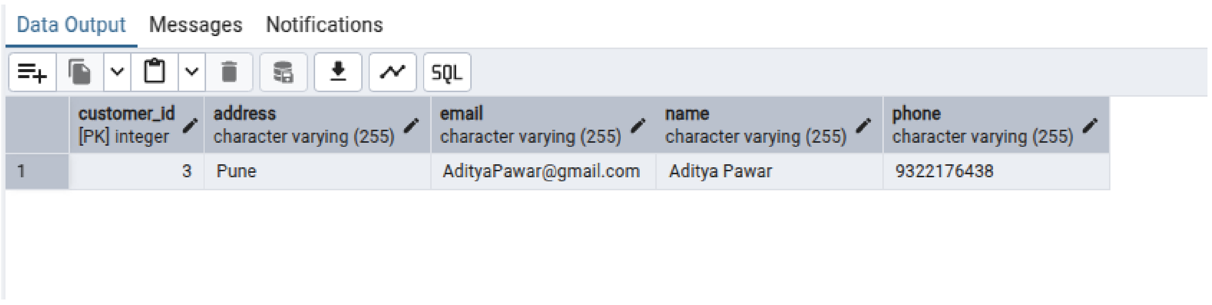
case 8 →



```
        i1_0.date,
        i1_0.notes,
        i1_0.product,
        i1_0.rating
    from
        interactions i1_0
    where
        i1_0.rating=?
Hibernate:
    select
        c1_0.customer_id,
        c1_0.address,
        c1_0.email,
        c1_0.name,
        c1_0.phone
    from
        customer c1_0
    where
        c1_0.customer_id=?
Interaction{interactionId=2, customer=Customer{customerId=3, name='Aditya Pawar', email='AdityaPawar@gmail.com', phone='9322176438', addre
2025-06-08T22:59:34.797+05:30  INFO 28188 --- [CustomerRelationshipManagement] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2025-06-08T22:59:34.801+05:30  INFO 28188 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2025-06-08T22:59:34.812+05:30  INFO 28188 --- [CustomerRelationshipManagement] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

Process finished with exit code 0
```



	customer_id integer	date date	interaction_id [PK] integer	rating integer	notes character varying (255)	product character varying (255)
1	3	2025-06-08	2	5	Amazing Build Quality	Laptop



	customer_id [PK] integer	address character varying (255)	email character varying (255)	name character varying (255)	phone character varying (255)
1	3	Pune	AdityaPawar@gmail.com	Aditya Pawar	9322176438

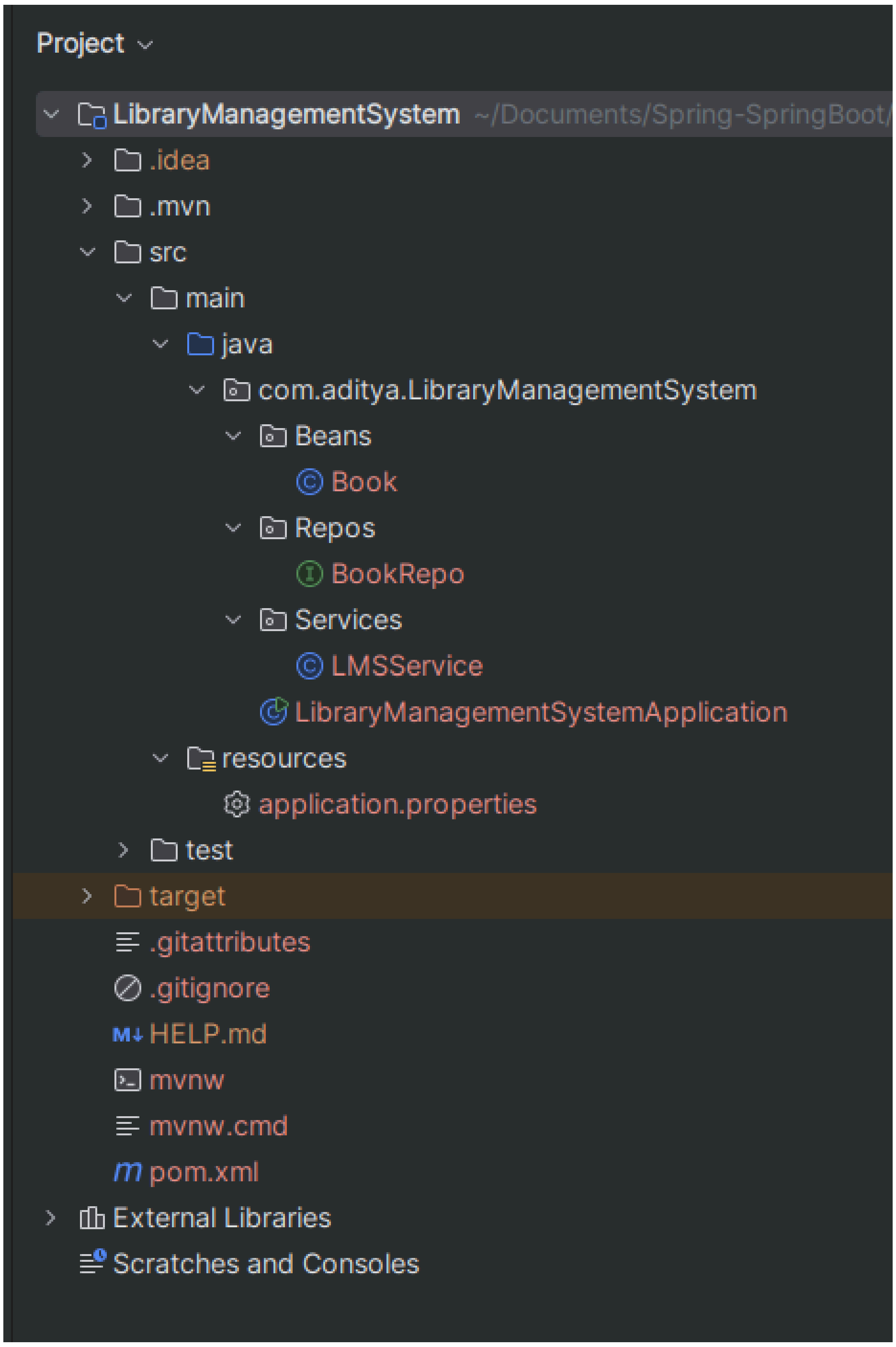
Que. 4 Library Management System Problem Statement:

Implement a system to manage books in a library. The system should facilitate:

- Create operations to add new books with details like title, author, ISBN, and publication year

- Read functionalities to retrieve book information and search for books by various criteria.
- Update capabilities to modify book details, such as changing the availability status or updating metadata.
- Delete operations to remove books that are no longer part of the collection,
- Fetch books by author, genre, or those published after a certain year.

Folder Structure:



LibraryManagementSystemApplication.java

```
package com.aditya.LibraryManagementSystem;

import com.aditya.LibraryManagementSystem.Beans.Book;
import com.aditya.LibraryManagementSystem.Services.LMSService;
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.Scanner;
import java.util.SimpleTimeZone;

@SpringBootApplication
public class LibraryManagementSystemApplication implements CommandLine

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

    @Autowired
    private LMSService lmsService;

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

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 1: Add new Book");
        System.out.println("Enter 2: Get Book ");
        System.out.println("Enter 3: Update Book");
        System.out.println("Enter 4: Delete Book");
        System.out.println("Enter 5: Fetch Books by Data");

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

        switch (operation)
        {
            case 1:
                System.out.println("Enter Book Title");
                String bookTitle = sc.nextLine();
                System.out.println("Enter Book Author");
                String bookAuthor = sc.nextLine();
                System.out.println("Enter Book Genre");
                String bookGenre = sc.nextLine();
                System.out.println("Enter Book ISBN Number");
                String bookISBN = sc.nextLine();
                System.out.println("Enter Book Publication Year");
                int bookYear = sc.nextInt();
                sc.nextLine();
                System.out.println("Is Book Available? (y/n)");
                String bookAvailability = sc.nextLine().trim().toLowerCase();
                boolean bookAvailable = bookAvailability.equals("y") ||
                bookAvailability.equals("yes");

                Book book = new Book();
                book.setTitle(bookTitle);
                book.setAuthor(bookAuthor);
                book.setGenre(bookGenre);
                book.setIsbn(bookISBN);
                book.setPublicationYear(bookYear);
                book.setAvailable(bookAvailable);

                lmsService.addBook(book);
                break;

            case 2:
                System.out.println("Enter Book ID");
                int bookId = sc.nextInt();
                sc.nextLine();
                Book book1 = lmsService.getBookById(bookId);
                System.out.println(book1.toString());
                break;

            case 3:
                System.out.print("Enter Book ID to update: ");
                int updateId = sc.nextInt();
                sc.nextLine();
                Book updateBook = lmsService.getBookById(updateId);

                if (updateBook != null) {
                    System.out.print("Is the book available? (true/false): ");
                    boolean avail = sc.nextBoolean();
                    updateBook.setAvailable(avail);
                    lmsService.updateBook(updateBook);
                    System.out.println("Book updated successfully.");
                } else {
                    System.out.println("Book not found.");
                }
                break;

            case 4:
                System.out.print("Enter Book ID to delete: ");
                int deleteId = sc.nextInt();
                sc.nextLine();
                lmsService.deleteBook(deleteId);
                System.out.println("Book deleted successfully.");
                break;

            case 5:
                System.out.println("1. By Author\n2. By Genre\n3. After Year");
                int opt = sc.nextInt();
                sc.nextLine();

                if (opt == 1) {
                    System.out.print("Enter Author: ");
                    String auth = sc.nextLine();
                    lmsService.getBooksByAuthor(auth).forEach(System.out::println);
                } else if (opt == 2) {
                    System.out.print("Enter Genre: ");
                    String gen = sc.nextLine();
                    lmsService.getBooksByGenre(gen).forEach(System.out::println);
                } else if (opt == 3) {
                    System.out.print("Enter Year: ");
                    int yr = sc.nextInt();
                    lmsService.getBooksAfterYear(yr).forEach(System.out::println);
                } else {
                    System.out.println("Invalid Option");
                }
                break;
```

```
            default:
                System.err.println("Invalid Input...!!!");
        }
    }
}
```

Book.java

```
package com.aditya.LibraryManagementSystem.Beans;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@NoArgsConstructor
@Entity
@Table(name = "books")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private @Getter @Setter int bookId;

    private @Getter @Setter String title;
    private @Getter @Setter String author;
    private @Getter @Setter String genre;
    private @Getter @Setter String isbn;
    private @Getter @Setter int publicationYear;
    private @Getter @Setter boolean available = true;

    public Book(boolean available, int publicationYear, String isbn, String genre,
        this.available = available;
        this.publicationYear = publicationYear;
        this.isbn = isbn;
        this.genre = genre;
        this.author = author;
        this.title = title;
        this.bookId = bookId;
    }

    @Override
    public String toString() {
        return "Book{" +
                "bookId=" + bookId +
                ", title='" + title + '\'' +
                ", author='" + author + '\'' +
                ", genre='" + genre + '\'' +
                ", isbn='" + isbn + '\'' +
                ", publicationYear=" + publicationYear +
                ", available=" + available +
                '}';
    }
}
```

LMSService.java

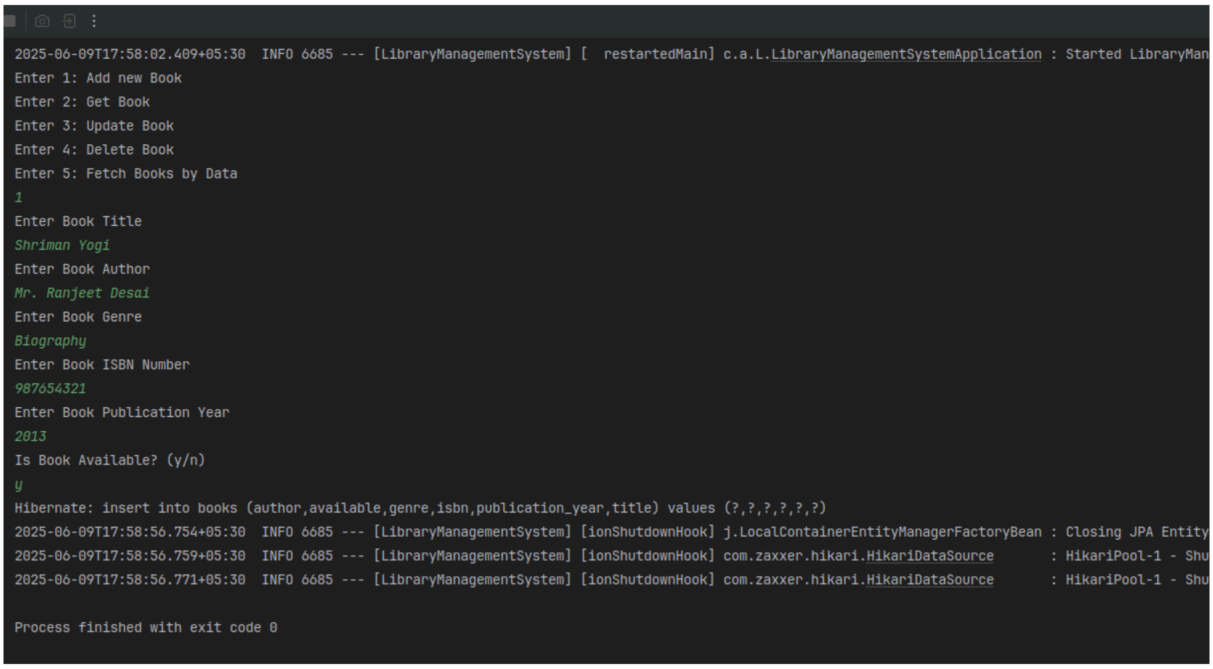
```
package com.aditya.LibraryManagementSystem.Services;

import com.aditya.LibraryManagementSystem.Beans.Book;
import com.aditya.LibraryManagementSystem.Repos.BookRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class LMSService {

    @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);
    }

    public List<Book> getBooksByAuthor(String author) {
        return bookRepo.findByAuthor(author);
    }

    public List<Book> getBooksByGenre(String genre) {
        return bookRepo.findByGenre(genre);
    }

    public List<Book> getBooksAfterYear(int year) {
        return bookRepo.findByPublicationYearGreaterThan(year);
    }
}
```

BookRepo.java

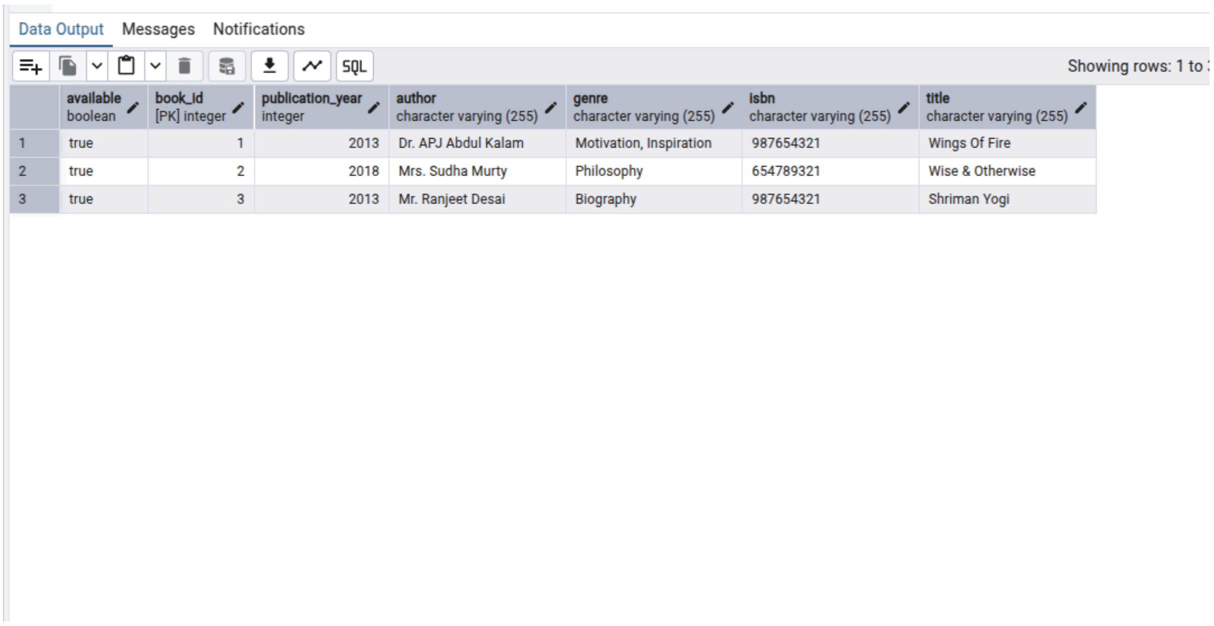
```
package com.aditya.LibraryManagementSystem.Repos;

import com.aditya.LibraryManagementSystem.Beans.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

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

    public List<Book> findByAuthor(String author);

    public List<Book> findByGenre(String genre);

    public List<Book> findByPublicationYearGreaterThan(int year);
}
```

Output:

case 1 →



```
2025-06-09T17:58:02.409+05:30  INFO 6685 --- [LibraryManagementSystem] [  restartedMain] c.a.L.LibraryManagementSystemApplication : Started LibraryMan
Enter 1: Add new Book
Enter 2: Get Book
Enter 3: Update Book
Enter 4: Delete Book
Enter 5: Fetch Books by Data
1
Enter Book Title
Shriman Yogi
Enter Book Author
Mr. Ranjeet Desai
Enter Book Genre
Biography
Enter Book ISBN Number
987654321
Enter Book Publication Year
2013
Is Book Available? (y/n)
y
Hibernate: insert into books (author,available,genre,isbn,publication_year,title) values (?,?,?,?,?,?)
2025-06-09T17:58:56.754+05:30  INFO 6685 --- [LibraryManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Entity
2025-06-09T17:58:56.759+05:30  INFO 6685 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu
2025-06-09T17:58:56.771+05:30  INFO 6685 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu

Process finished with exit code 0
```

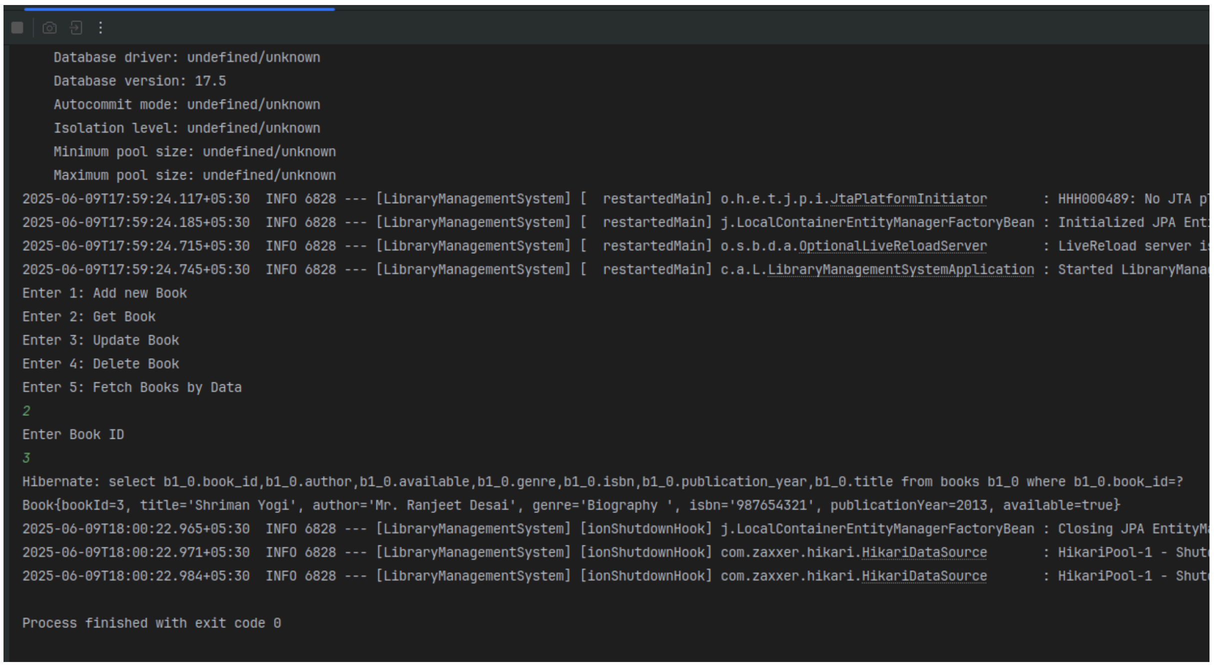


Data Output Messages Notifications

Showing rows: 1 to

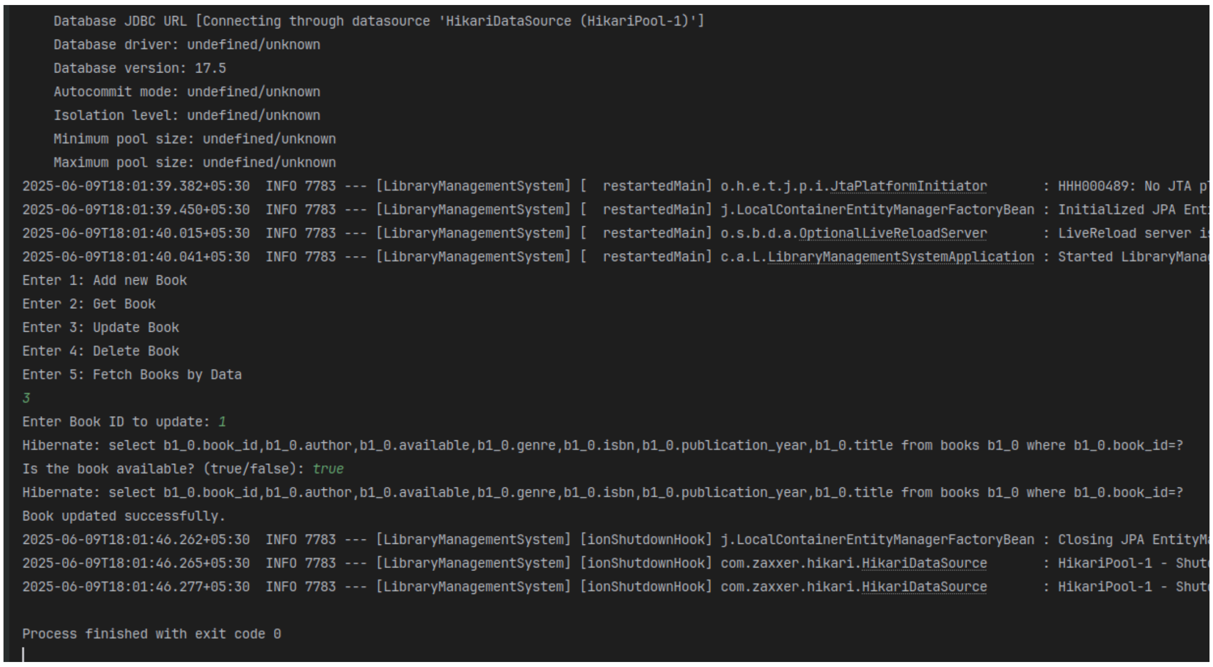
	available boolean	book_id [PK] integer	publication_year integer	author character varying (255)	genre character varying (255)	isbn character varying (255)	title character varying (255)
1	true	1	2013	Dr. APJ Abdul Kalam	Motivation, Inspiration	987654321	Wings Of Fire
2	true	2	2018	Mrs. Sudha Murty	Philosophy	654789321	Wise & Otherwise
3	true	3	2013	Mr. Ranjeet Desai	Biography	987654321	Shriman Yogi

case 2 →



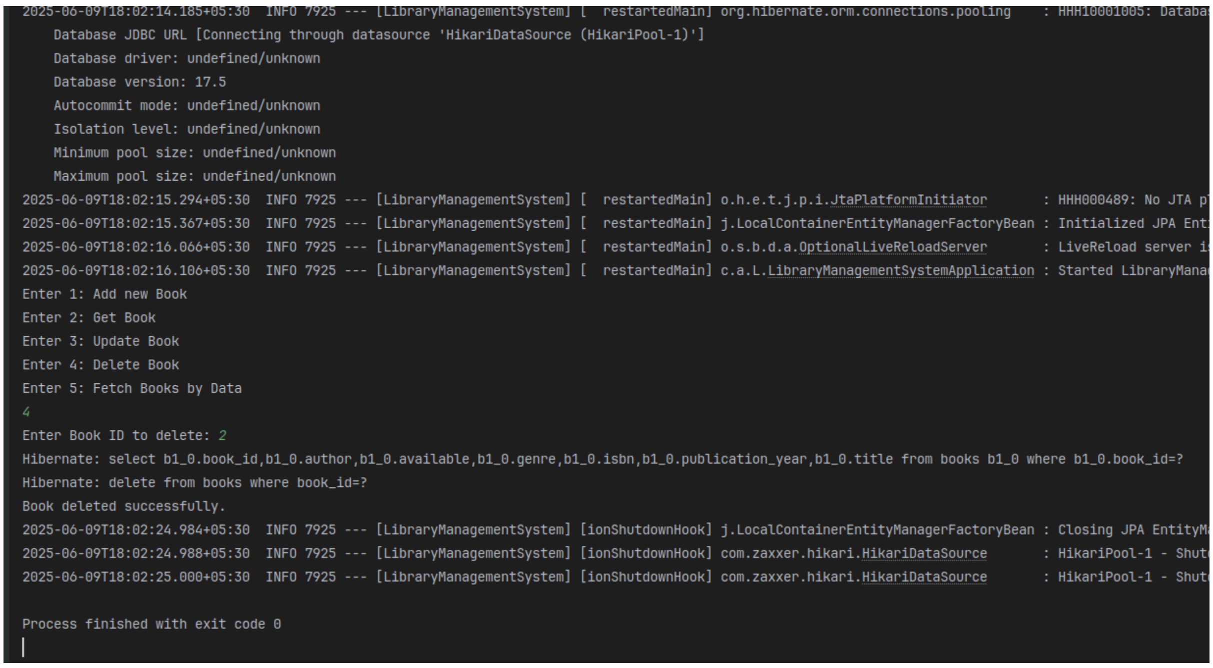
```
    Database driver: undefined/unknown
    Database version: 17.5
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T17:59:24.117+05:30  INFO 6828 --- [LibraryManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA p
2025-06-09T17:59:24.185+05:30  INFO 6828 --- [LibraryManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA Ent
2025-06-09T17:59:24.715+05:30  INFO 6828 --- [LibraryManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server i
2025-06-09T17:59:24.745+05:30  INFO 6828 --- [LibraryManagementSystem] [  restartedMain] c.a.L.LibraryManagementSystemApplication : Started LibraryMana
Enter 1: Add new Book
Enter 2: Get Book
Enter 3: Update Book
Enter 4: Delete Book
Enter 5: Fetch Books by Data
2
Enter Book ID
3
Hibernate: select b1_0.book_id,b1_0.author,b1_0.available,b1_0.genre,b1_0.isbn,b1_0.publication_year,b1_0.title from books b1_0 where b1_0.book_id=?
Book{bookId=3, title='Shriman Yogi', author='Mr. Ranjeet Desai', genre='Biography ', isbn='987654321', publicationYear=2013, available=true}
2025-06-09T18:00:22.965+05:30  INFO 6828 --- [LibraryManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityM
2025-06-09T18:00:22.971+05:30  INFO 6828 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shut
2025-06-09T18:00:22.984+05:30  INFO 6828 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shut

Process finished with exit code 0
```

case 3 →



```
    Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
    Database driver: undefined/unknown
    Database version: 17.5
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T18:01:39.382+05:30  INFO 7783 --- [LibraryManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA p
2025-06-09T18:01:39.450+05:30  INFO 7783 --- [LibraryManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA Ent
2025-06-09T18:01:40.015+05:30  INFO 7783 --- [LibraryManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server i
2025-06-09T18:01:40.041+05:30  INFO 7783 --- [LibraryManagementSystem] [  restartedMain] c.a.L.LibraryManagementSystemApplication : Started LibraryManag
Enter 1: Add new Book
Enter 2: Get Book
Enter 3: Update Book
Enter 4: Delete Book
Enter 5: Fetch Books by Data
3
Enter Book ID to update: 1
Hibernate: select b1_0.book_id,b1_0.author,b1_0.available,b1_0.genre,b1_0.isbn,b1_0.publication_year,b1_0.title from books b1_0 where b1_0.book_id=?
Is the book available? (true/false): true
Hibernate: select b1_0.book_id,b1_0.author,b1_0.available,b1_0.genre,b1_0.isbn,b1_0.publication_year,b1_0.title from books b1_0 where b1_0.book_id=?
Book updated successfully.
2025-06-09T18:01:46.262+05:30  INFO 7783 --- [LibraryManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMa
2025-06-09T18:01:46.265+05:30  INFO 7783 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shut
2025-06-09T18:01:46.277+05:30  INFO 7783 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shut

Process finished with exit code 0
```

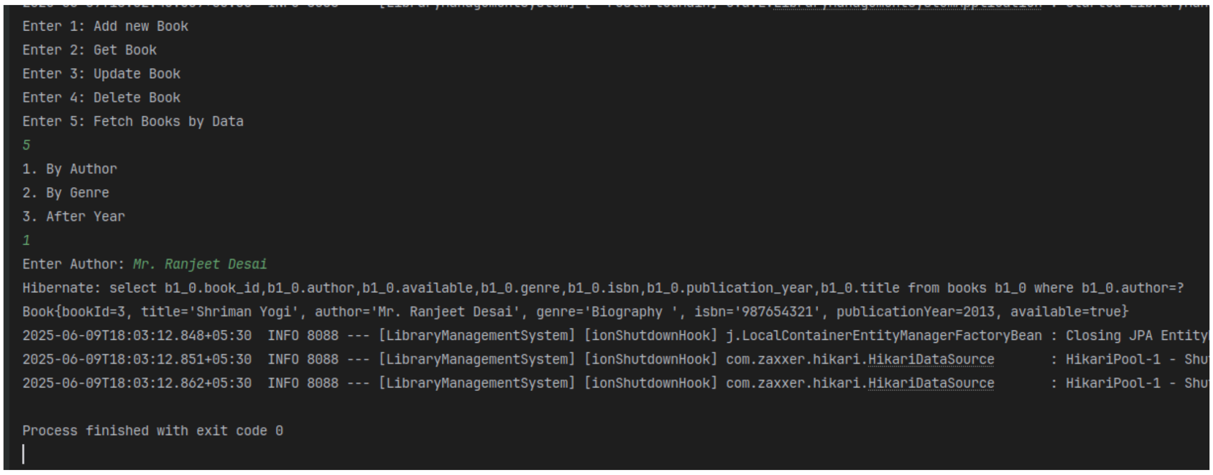
case 4 →



```
2025-06-09T18:02:14.185+05:30  INFO 7925 --- [LibraryManagementSystem] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Databa
    Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
    Database driver: undefined/unknown
    Database version: 17.5
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T18:02:15.294+05:30  INFO 7925 --- [LibraryManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA p
2025-06-09T18:02:15.367+05:30  INFO 7925 --- [LibraryManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA Ent
2025-06-09T18:02:16.066+05:30  INFO 7925 --- [LibraryManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server i
2025-06-09T18:02:16.106+05:30  INFO 7925 --- [LibraryManagementSystem] [  restartedMain] c.a.L.LibraryManagementSystemApplication : Started LibraryManag
Enter 1: Add new Book
Enter 2: Get Book
Enter 3: Update Book
Enter 4: Delete Book
Enter 5: Fetch Books by Data
4
Enter Book ID to delete: 2
Hibernate: select b1_0.book_id,b1_0.author,b1_0.available,b1_0.genre,b1_0.isbn,b1_0.publication_year,b1_0.title from books b1_0 where b1_0.book_id=?
Hibernate: delete from books where book_id=?
Book deleted successfully.
2025-06-09T18:02:24.984+05:30  INFO 7925 --- [LibraryManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMa
2025-06-09T18:02:24.988+05:30  INFO 7925 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shut
2025-06-09T18:02:25.000+05:30  INFO 7925 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shut

Process finished with exit code 0
```

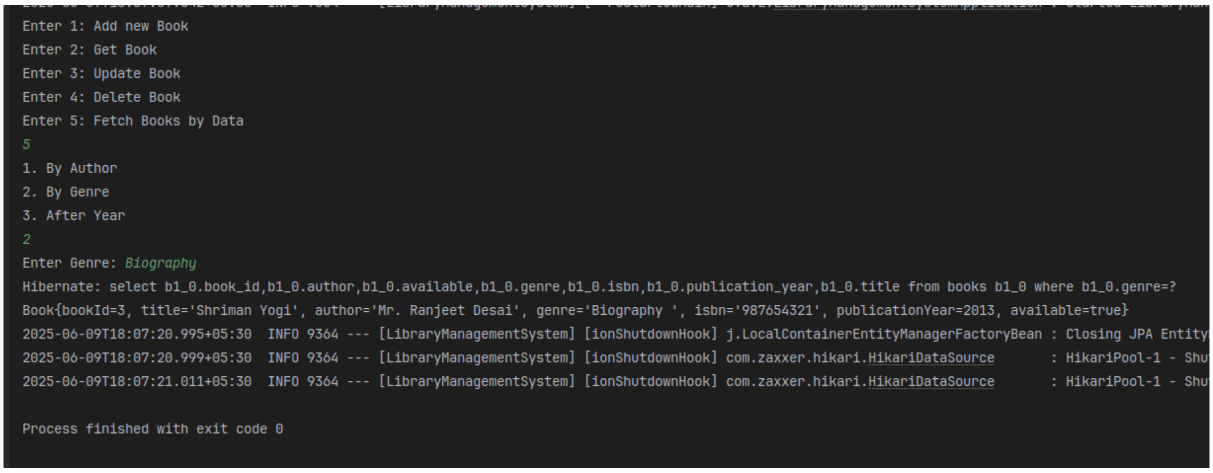
case 5 →

a. by Author



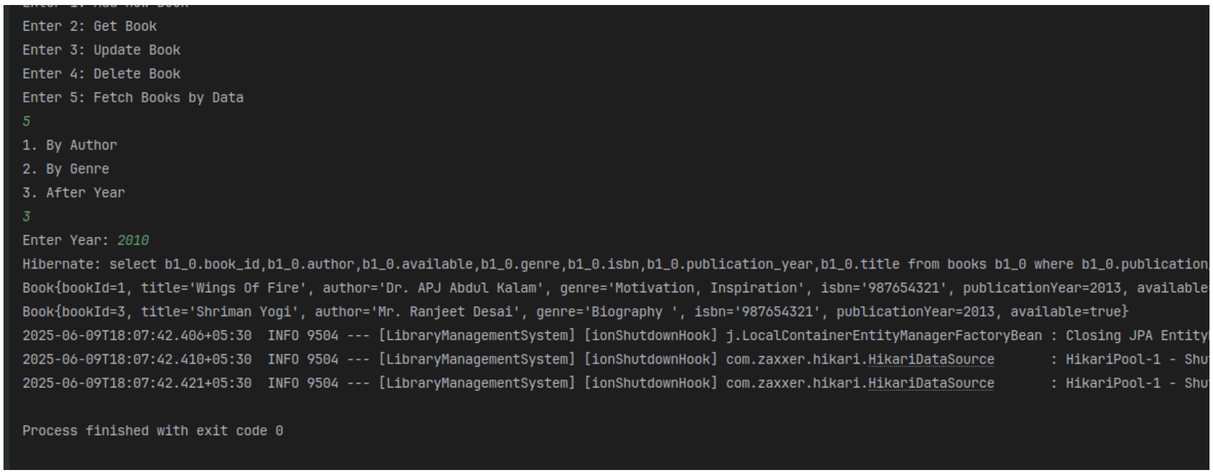
```
Enter 1: Add new Book
Enter 2: Get Book
Enter 3: Update Book
Enter 4: Delete Book
Enter 5: Fetch Books by Data
5
1. By Author
2. By Genre
3. After Year
1
Enter Author: Mr. Ranjeet Desai
Hibernate: select b1_0.book_id,b1_0.author,b1_0.available,b1_0.genre,b1_0.isbn,b1_0.publication_year,b1_0.title from books b1_0 where b1_0.author=?
Book{bookId=3, title='Shriman Yogi', author='Mr. Ranjeet Desai', genre='Biography ', isbn='987654321', publicationYear=2013, available=true}
2025-06-09T18:03:12.848+05:30  INFO 8088 --- [LibraryManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Entity
2025-06-09T18:03:12.851+05:30  INFO 8088 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu
2025-06-09T18:03:12.862+05:30  INFO 8088 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu

Process finished with exit code 0
```

b. by Genre



```
Enter 1: Add new Book
Enter 2: Get Book
Enter 3: Update Book
Enter 4: Delete Book
Enter 5: Fetch Books by Data
5
1. By Author
2. By Genre
3. After Year
2
Enter Genre: Biography
Hibernate: select b1_0.book_id,b1_0.author,b1_0.available,b1_0.genre,b1_0.isbn,b1_0.publication_year,b1_0.title from books b1_0 where b1_0.genre=?
Book{bookId=3, title='Shriman Yogi', author='Mr. Ranjeet Desai', genre='Biography ', isbn='987654321', publicationYear=2013, available=true}
2025-06-09T18:07:20.995+05:30  INFO 9364 --- [LibraryManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Entity
2025-06-09T18:07:20.999+05:30  INFO 9364 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu
2025-06-09T18:07:21.011+05:30  INFO 9364 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu

Process finished with exit code 0
```

c. by year



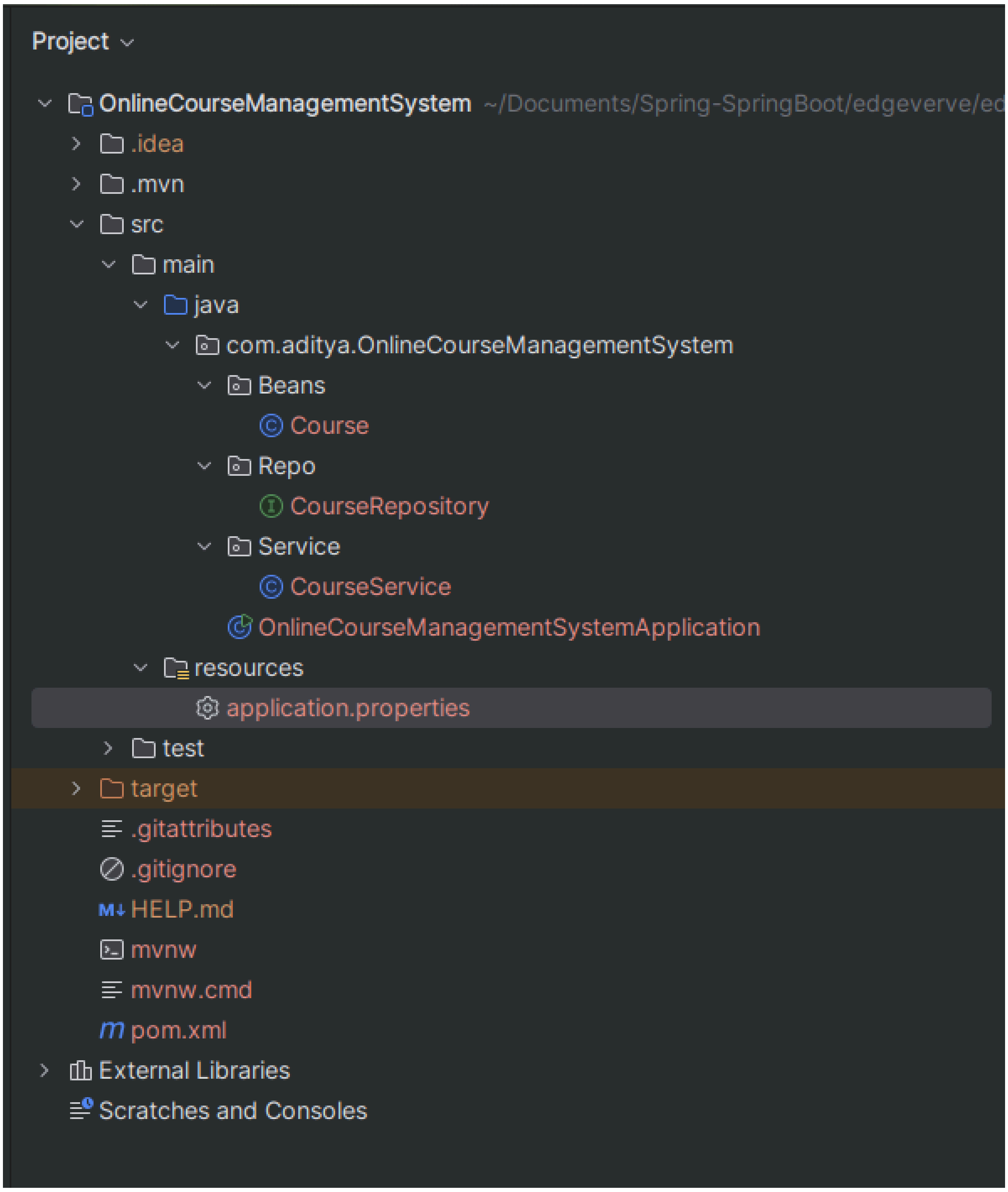
```
Enter 2: Get Book
Enter 3: Update Book
Enter 4: Delete Book
Enter 5: Fetch Books by Data
5
1. By Author
2. By Genre
3. After Year
3
Enter Year: 2010
Hibernate: select b1_0.book_id,b1_0.author,b1_0.available,b1_0.genre,b1_0.isbn,b1_0.publication_year,b1_0.title from books b1_0 where b1_0.publication
Book{bookId=1, title='Wings Of Fire', author='Dr. APJ Abdul Kalam', genre='Motivation, Inspiration', isbn='987654321', publicationYear=2013, available
Book{bookId=3, title='Shriman Yogi', author='Mr. Ranjeet Desai', genre='Biography ', isbn='987654321', publicationYear=2013, available=true}
2025-06-09T18:07:42.406+05:30  INFO 9504 --- [LibraryManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA Entity
2025-06-09T18:07:42.410+05:30  INFO 9504 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu
2025-06-09T18:07:42.421+05:30  INFO 9504 --- [LibraryManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shu

Process finished with exit code 0
```

Online Course Management System Problem Statement:

Create a system to manage online courses and student enrollments. The system should support:

- Create operations to add new courses with details like title, description, instructor, and schedule
- Read functionalities to view course information and list all available courses.
- Update capabilities to modify course details or update schedules.
- Delete operations to remove courses that are no longer offered.
- Fetch courses by instructor, category, or those scheduled within a specific time frame.

Folder Structure:



```
Project
OnlineCourseManagementSystem ~/Documents/Spring-SpringBoot/edgeverve/ed
  .idea
  .mvn
  src
    main
      java
        com.aditya.OnlineCourseManagementSystem
          Beans
            Course
          Repo
            CourseRepository
          Service
            CourseService
          OnlineCourseManagementSystemApplication
      resources
        application.properties
    test
  target
  .gitattributes
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

OnlineCourseManagementApplication.java

```java
package com.aditya.OnlineCourseManagementSystem;

import com.aditya.OnlineCourseManagementSystem.Beans.Course;
import com.aditya.OnlineCourseManagementSystem.Service.CourseService;
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.time.LocalDate;
import java.util.List;
import java.util.Scanner;

@SpringBootApplication
public class OnlineCourseManagementSystemApplication implements
CommandLineRunner {

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

    @Autowired
    private CourseService courseService;

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

        System.out.println("Enter 1: Add Course");
        System.out.println("Enter 2: Show Course By:");
        System.out.println("Enter 3: Update Course Details");
        System.out.println("Enter 4: Delete course");
        System.out.println("Enter 5: Fetch Courses By:");

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

        switch (operation)
        {
            case 1:
                System.out.println("Enter Course Title");
                String title = sc.nextLine();
                System.out.println("Enter Course Description");
                String desc = sc.nextLine();
                System.out.println("Enter Course Instructor Name");
                String intru = sc.nextLine();
                System.out.println("Enter Course Category");
                String categ = sc.nextLine();
                System.out.println("Enter Course Start Date (yyyy-mm-dd): ");
                LocalDate startDate = LocalDate.parse(sc.nextLine());
                System.out.println("Enter Course End Date (yyyy-mm-dd):");
                LocalDate endDate = LocalDate.parse(sc.nextLine());

                Course course = new Course();
                course.setTitle(title);
                course.setDescription(desc);
                course.setInstructor(intru);
                course.setCategory(categ);
                course.setStartDate(startDate);
                course.setEndDate(endDate);
                courseService.addCourse(course);
                break;

            case 2:
                System.out.println("1: Course ID \n2: List All Course");
                int option = sc.nextInt();

                switch (option) {
                    case 1:
                        System.out.println("Enter Course ID");
                        int id = sc.nextInt();
                        sc.nextLine();
                        Course c1 =  courseService.getCourseById(id);
                        System.out.println(c1.toString());
                        break;

                    case 2:
                        List<Course> courseList = courseService.getAllCourses();
                        courseList.forEach(course1 -> {
                            System.out.println(course1);
                        });
                        break;
                    default:
                        System.err.println("Enter valid Option!!!");
                }
            break;

            case 3:
                System.out.println("Enter Course ID to Update:");
                int updateId = sc.nextInt();
                sc.nextLine();
                try {
                    Course existing = courseService.getCourseById(updateId);
                    if (existing != null) {
                        System.out.println("New Title:");
                        existing.setTitle(sc.nextLine());
                        System.out.println("New Description:");
                        existing.setDescription(sc.nextLine());
                        System.out.println("New Instructor:");
                        existing.setInstructor(sc.nextLine());
                        System.out.println("New Category:");
                        existing.setCategory(sc.nextLine());
                        System.out.println("New Start Date:");
                        existing.setStartDate(LocalDate.parse(sc.nextLine()));
                        System.out.println("New End Date:");
                        existing.setEndDate(LocalDate.parse(sc.nextLine()));
                        courseService.updateCourse(existing);
                    } else {
                        System.out.println("Course not found");
                    }
                } catch (Exception e) {
                    System.err.println("Invalid ID");
                }
                break;

            case 4:
                System.out.println("Enter ID to Delete:");
                courseService.deleteCourse(sc.nextInt());
                break;
            case 5:
                System.out.println("1: Instructor \n2:Category \n3:Date Range ");
                int choice = sc.nextInt();
                sc.nextLine();

                switch (choice) {

                    case 1:
                        System.out.println("Enter Instructor Name:");
                        courseService.fetchByInstructor(sc.nextLine()).forEach(System
                        println);
                        break;

                    case 2:
                        System.out.println("Enter Category:");
                        courseService.fetchByCategory(sc.nextLine()).forEach(System.
                        println);
                        break;

                    case 3:
                        System.out.println("Enter Start Date (yyyy-mm-dd):");
                        LocalDate sDate = LocalDate.parse(sc.nextLine());
                        System.out.println("Enter End Date (yyyy-mm-dd):");
                        LocalDate eDate = LocalDate.parse(sc.nextLine());
                        courseService.fetchBySchedule(sDate, eDate).forEach(System.
                        println);
                        break;

                    default:
                        System.err.println("Invalid Choice!!!");
                }
                break;

            default:
                System.err.println("Invalid Input...!");
        }
        }
}
```

Course.java

```
package com.aditya.OnlineCourseManagementSystem.Beans;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.time.LocalDate;

@NoArgsConstructor
@Entity
@Table(name = "courses")
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private @Getter @Setter int courseId;
    @NotBlank(message = "Title cannot be blank")
    private @Getter @Setter String title;
    @Column(columnDefinition = "TEXT")
    private @Getter @Setter String description;
    @NotBlank(message = "Instructor Name Required")
    private @Getter @Setter String instructor;
    private @Getter @Setter String category;

    private @Getter @Setter LocalDate startDate;
    private @Getter @Setter LocalDate endDate;

    public Course(int courseId, String title, String category, String
    instructor, String description, LocalDate startDate, LocalDate endDate)
    {
        this.courseId = courseId;
        this.title = title;
        this.category = category;
        this.instructor = instructor;
        this.description = description;
        this.endDate = endDate;
        this.startDate = startDate;
    }

    @Override
    public String toString() {
        return "Course{" +
                "courseId=" + courseId +
                ", title='" + title + '\'' +
                ", description='" + description + '\'' +
                ", instructor='" + instructor + '\'' +
                ", category='" + category + '\'' +
                ", start=" + startDate +
                ", endDate=" + endDate +
                '}';
    }
}
```

CourseRepository.java

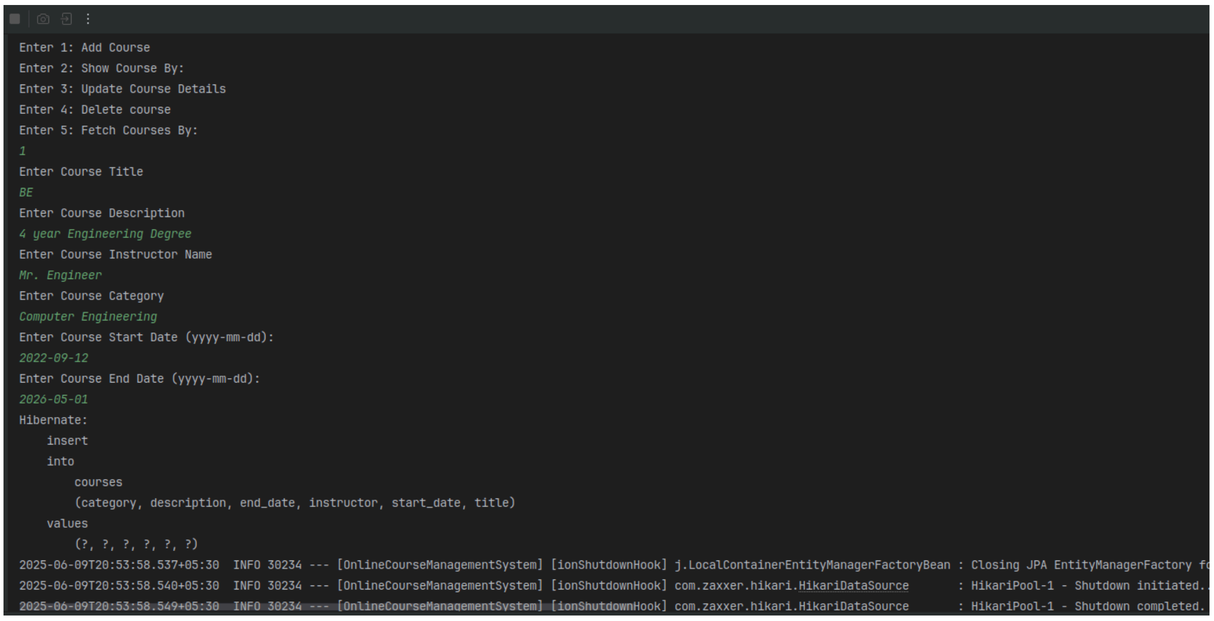
```
package com.aditya.OnlineCourseManagementSystem.Repo;

import com.aditya.OnlineCourseManagementSystem.Beans.Course;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

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

@Repository
public interface CourseRepository extends JpaRepository<Course,Integer> {

    public List<Course> findByCategory(String category);

    public List<Course> findByInstructor(String instructor);

    public List<Course> findByStartDateGreaterThanEqualAndEndDateLess
    ThanEqual(LocalDate startDate, LocalDate endDate);
}
```

CourseService.java

```
package com.aditya.OnlineCourseManagementSystem.Service;

import com.aditya.OnlineCourseManagementSystem.Beans.Course;
import com.aditya.OnlineCourseManagementSystem.Repo.CourseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

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

@Service
public class CourseService {

    @Autowired
    private CourseRepository courseRepository;

    public void addCourse(Course course){
        courseRepository.save(course);
    }

    public Course getCourseById(int courseId){
        return courseRepository.findById(courseId).orElse(null);
    }

    public List<Course> getAllCourses(){
        return courseRepository.findAll();
    }
    public void updateCourse(Course course){
        courseRepository.save(course);
    }

    public void deleteCourse(int id){
        courseRepository.deleteById(id);
    }

    public List<Course> fetchByInstructor(String instructor){
        return courseRepository.findByInstructor(instructor);
    }
    public List<Course> fetchByCategory(String category){
        return courseRepository.findByCategory(category);
    }
    public List<Course> fetchBySchedule(LocalDate from, LocalDate to){
        return courseRepository.findByStartDateGreaterThanEqualAndEndDateLe
    }
}
```

Output:

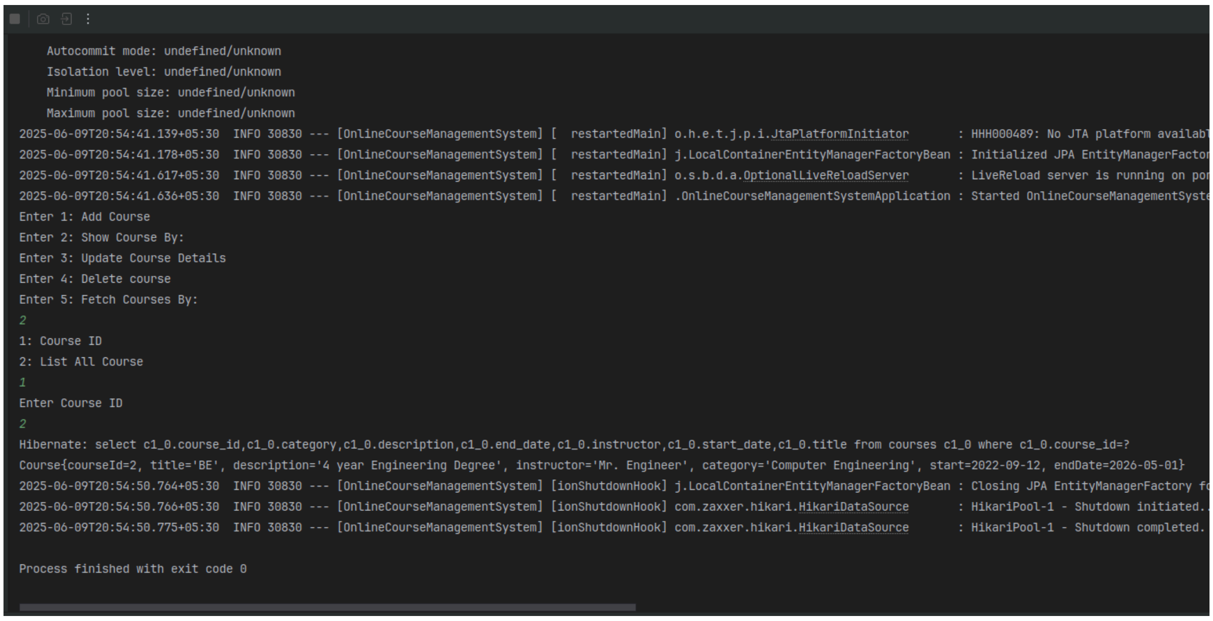
case 1 →



```
Enter 1: Add Course
Enter 2: Show Course By:
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
1
Enter Course Title
BE
Enter Course Description
4 year Engineering Degree
Enter Course Instructor Name
Mr. Engineer
Enter Course Category
Computer Engineering
Enter Course Start Date (yyyy-mm-dd):
2022-09-12
Enter Course End Date (yyyy-mm-dd):
2026-05-01
Hibernate:
    insert
    into
        courses
        (category, description, end_date, instructor, start_date, title)
    values
        (?, ?, ?, ?, ?, ?)
2025-06-09T20:53:58.537+05:30  INFO 30234 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory fo
2025-06-09T20:53:58.540+05:30  INFO 30234 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated..
2025-06-09T20:53:58.549+05:30  INFO 30234 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

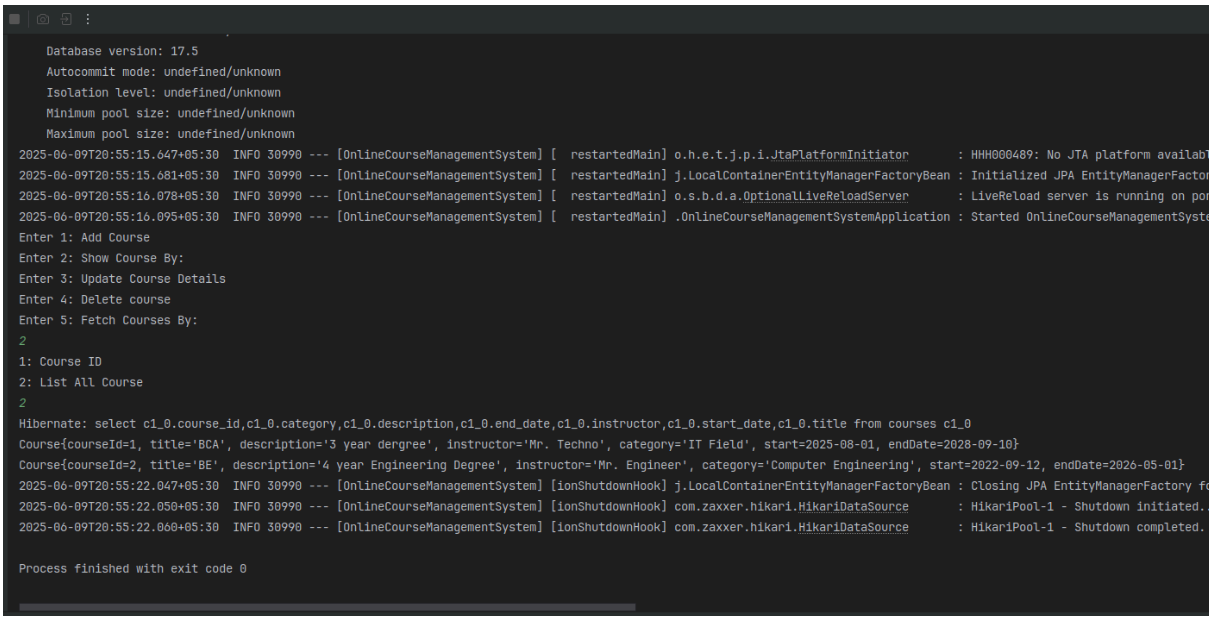
case 2 →

a. by course ID



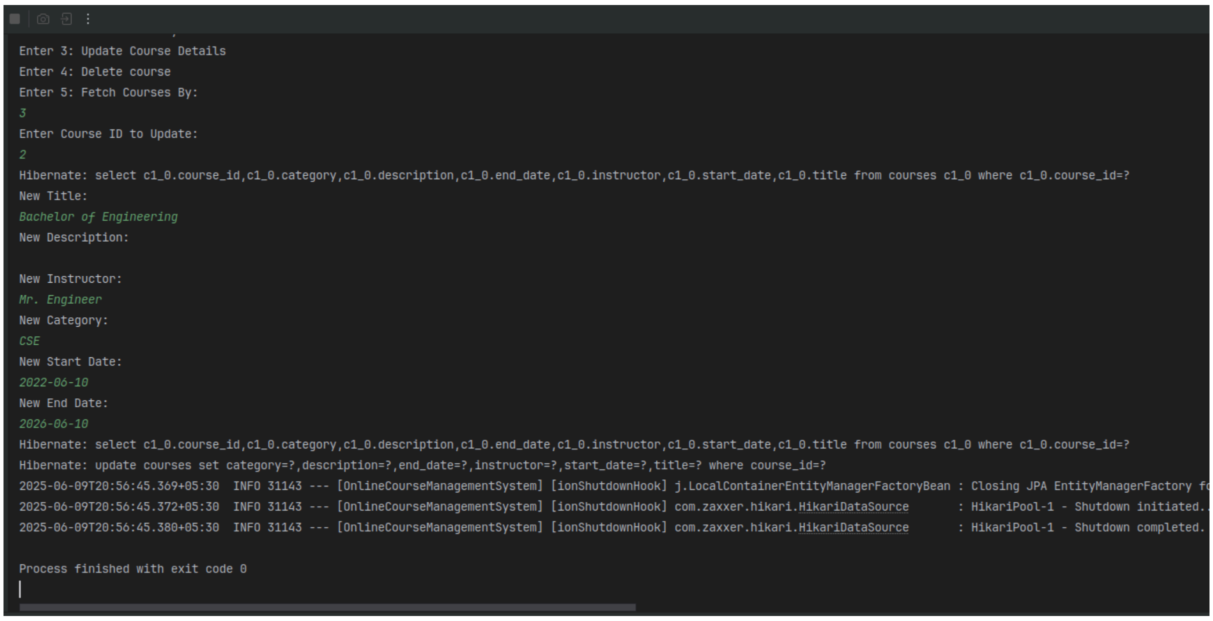
```
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T20:54:41.139+05:30  INFO 30830 --- [OnlineCourseManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform availab
2025-06-09T20:54:41.178+05:30  INFO 30830 --- [OnlineCourseManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
2025-06-09T20:54:41.617+05:30  INFO 30830 --- [OnlineCourseManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on po
2025-06-09T20:54:41.636+05:30  INFO 30830 --- [OnlineCourseManagementSystem] [  restartedMain] .OnlineCourseManagementSystemApplication : Started OnlineCourseManagementSyst
Enter 1: Add Course
Enter 2: Show Course By:
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
2
1: Course ID
2: List All Course
1
Enter Course ID
2
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0 where c1_0.course_id=?
Course{courseId=2, title='BE', description='4 year Engineering Degree', instructor='Mr. Engineer', category='Computer Engineering', start=2022-09-12, endDate=2026-05-01}
2025-06-09T20:54:50.764+05:30  INFO 30830 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2025-06-09T20:54:50.766+05:30  INFO 30830 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated.
2025-06-09T20:54:50.775+05:30  INFO 30830 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.

Process finished with exit code 0
```

b. List of Courses

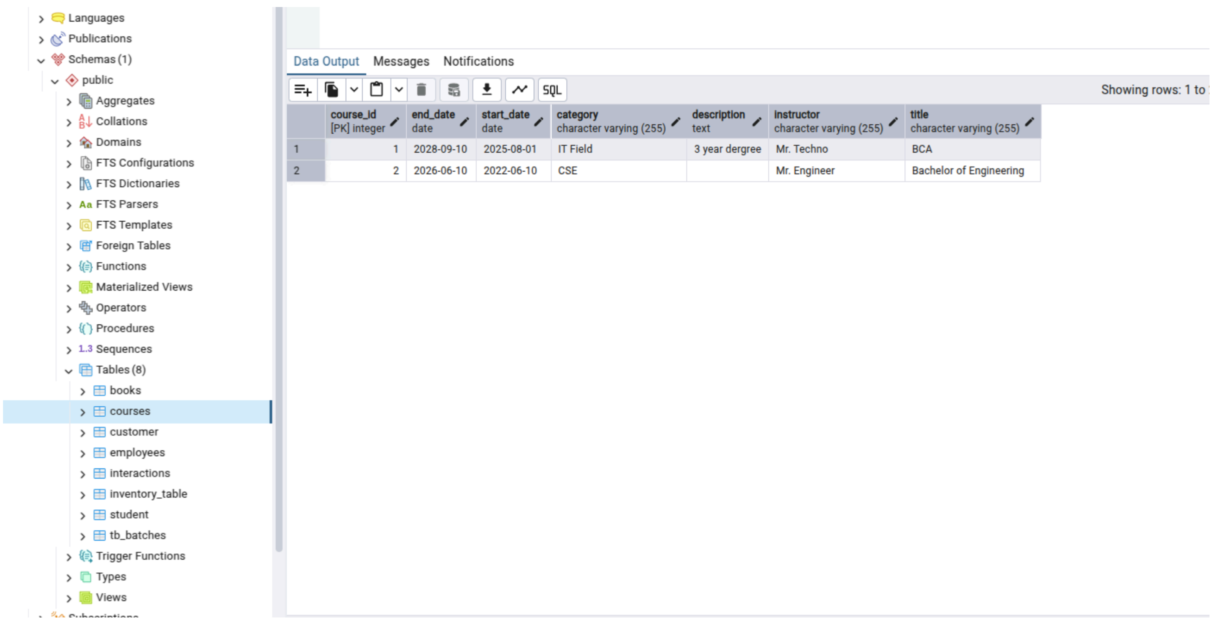


```
    Database version: 17.5
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T20:55:15.647+05:30  INFO 30990 --- [OnlineCourseManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform availab
2025-06-09T20:55:15.681+05:30  INFO 30990 --- [OnlineCourseManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
2025-06-09T20:55:16.078+05:30  INFO 30990 --- [OnlineCourseManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on po
2025-06-09T20:55:16.095+05:30  INFO 30990 --- [OnlineCourseManagementSystem] [  restartedMain] .OnlineCourseManagementSystemApplication : Started OnlineCourseManagementSyst
Enter 1: Add Course
Enter 2: Show Course By:
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
2
1: Course ID
2: List All Course
2
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0
Course{courseId=1, title='BCA', description='3 year dergree', instructor='Mr. Techno', category='IT Field', start=2025-08-01, endDate=2028-09-10}
Course{courseId=2, title='BE', description='4 year Engineering Degree', instructor='Mr. Engineer', category='Computer Engineering', start=2022-09-12, endDate=2026-05-01}
2025-06-09T20:55:22.047+05:30  INFO 30990 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2025-06-09T20:55:22.050+05:30  INFO 30990 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated.
2025-06-09T20:55:22.060+05:30  INFO 30990 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.

Process finished with exit code 0
```

case 3 →

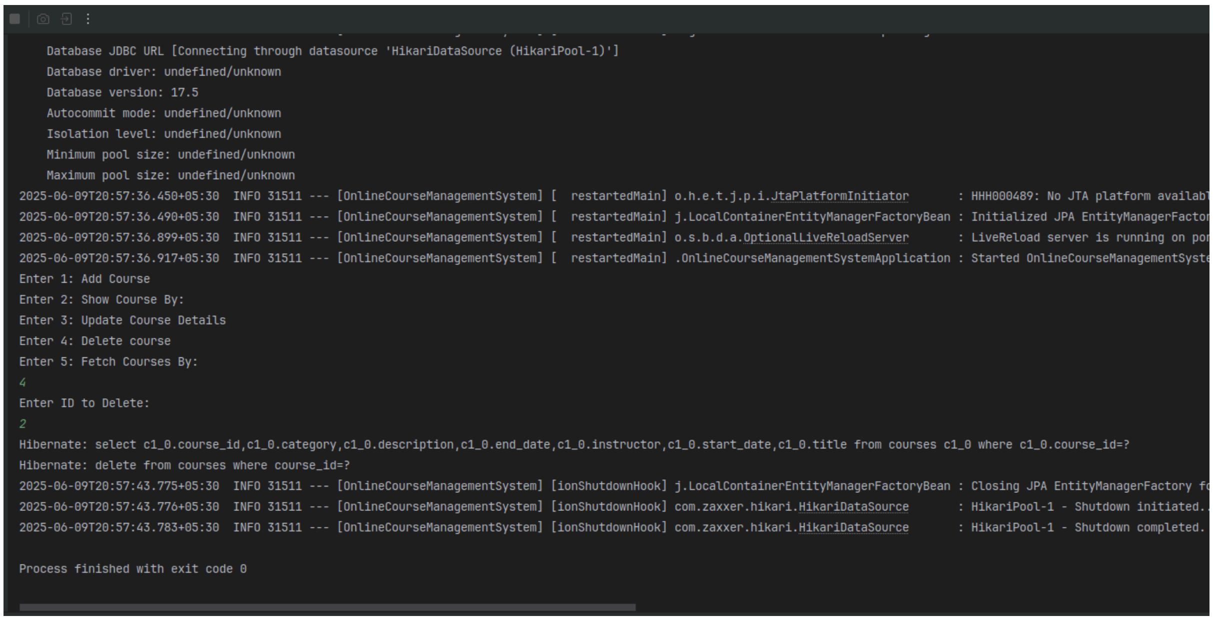


```
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
3
Enter Course ID to Update:
2
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0 where c1_0.course_id=?
New Title:
Bachelor of Engineering
New Description:

New Instructor:
Mr. Engineer
New Category:
CSE
New Start Date:
2022-06-10
New End Date:
2026-06-10
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0 where c1_0.course_id=?
Hibernate: update courses set category=?,description=?,end_date=?,instructor=?,start_date=?,title=? where course_id=?
2025-06-09T20:56:45.369+05:30  INFO 31143 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory fo
2025-06-09T20:56:45.372+05:30  INFO 31143 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated.
2025-06-09T20:56:45.380+05:30  INFO 31143 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.

Process finished with exit code 0
```



	course_id [PK] integer	end_date date	start_date date	category character varying (255)	description text	instructor character varying (255)	title character varying (255)
1	1	2028-09-10	2025-08-01	IT Field	3 year dergree	Mr. Techno	BCA
2	2	2026-06-10	2022-06-10	CSE		Mr. Engineer	Bachelor of Engineering

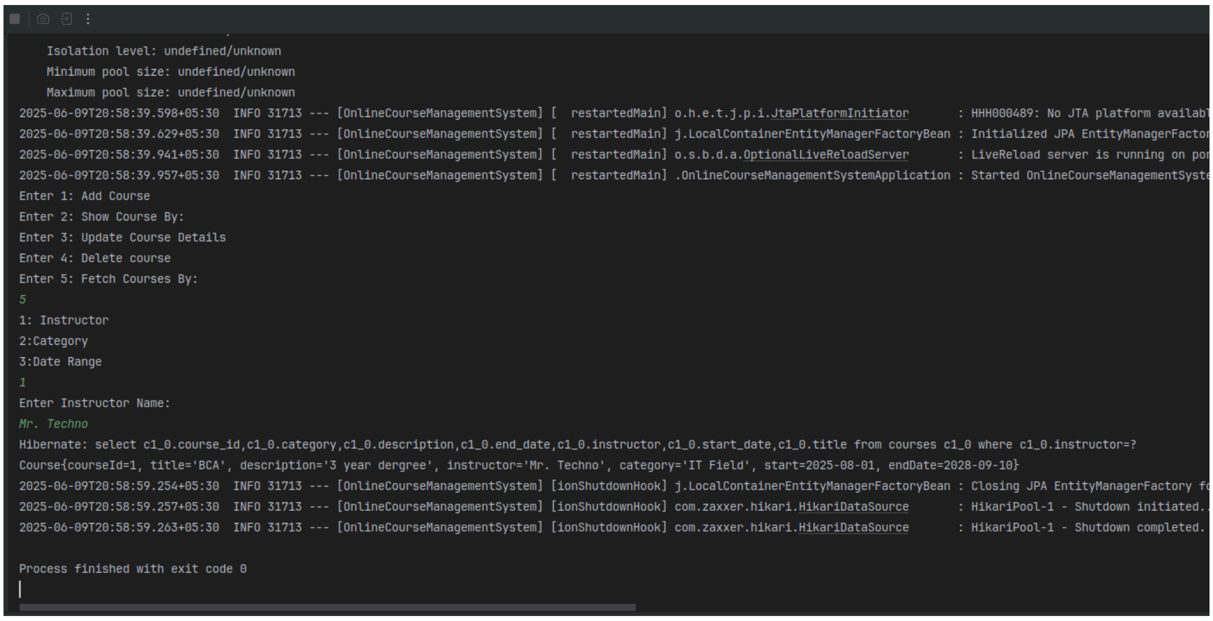
case 4 →



```
    Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
    Database driver: undefined/unknown
    Database version: 17.5
    Autocommit mode: undefined/unknown
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T20:57:36.450+05:30  INFO 31511 --- [OnlineCourseManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator      : HHH000489: No JTA platform availab
2025-06-09T20:57:36.490+05:30  INFO 31511 --- [OnlineCourseManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
2025-06-09T20:57:36.899+05:30  INFO 31511 --- [OnlineCourseManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer      : LiveReload server is running on po
2025-06-09T20:57:36.917+05:30  INFO 31511 --- [OnlineCourseManagementSystem] [  restartedMain] .OnlineCourseManagementSystemApplication : Started OnlineCourseManagementSyst
Enter 1: Add Course
Enter 2: Show Course By:
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
4
Enter ID to Delete:
2
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0 where c1_0.course_id=?
Hibernate: delete from courses where course_id=?
2025-06-09T20:57:43.775+05:30  INFO 31511 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2025-06-09T20:57:43.776+05:30  INFO 31511 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 - Shutdown initiated.
2025-06-09T20:57:43.783+05:30  INFO 31511 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 - Shutdown completed.

Process finished with exit code 0
```

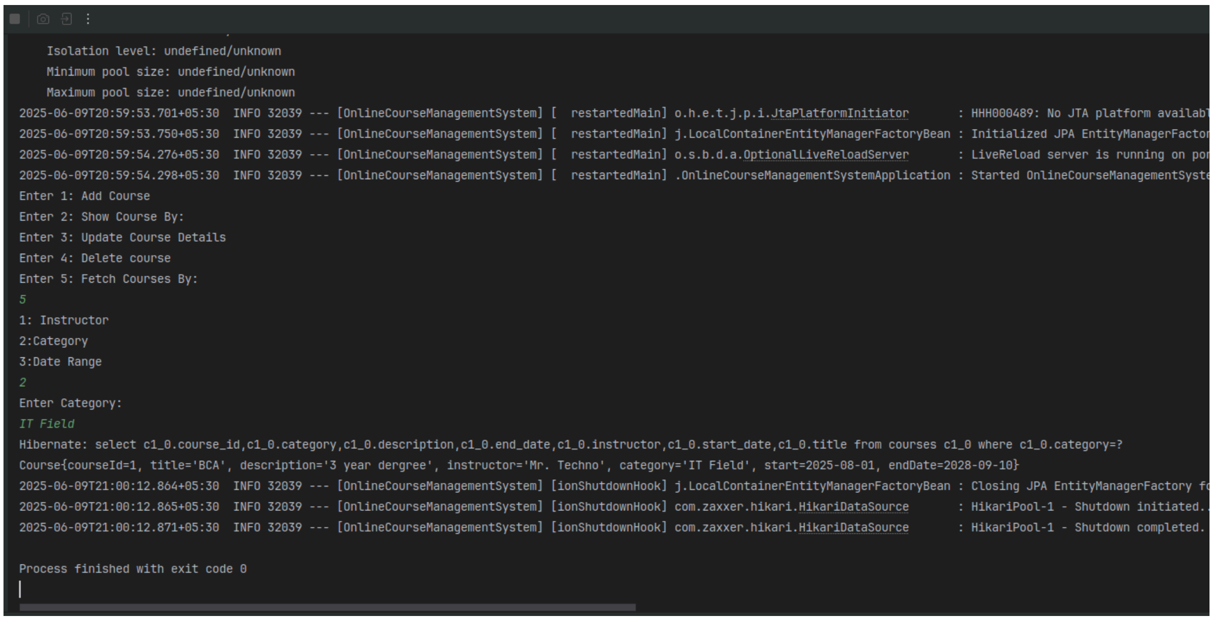
case 5 →

a. by Instructor



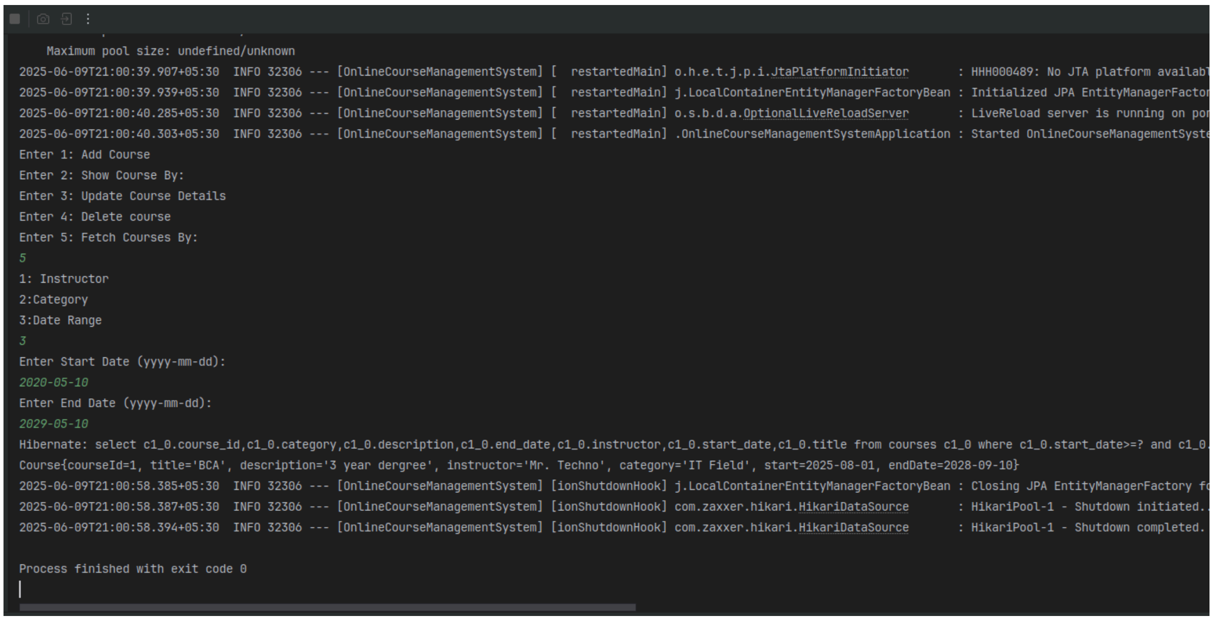
```
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T20:58:39.598+05:30  INFO 31713 --- [OnlineCourseManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator      : HHH000489: No JTA platform availab
2025-06-09T20:58:39.629+05:30  INFO 31713 --- [OnlineCourseManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
2025-06-09T20:58:39.941+05:30  INFO 31713 --- [OnlineCourseManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer      : LiveReload server is running on po
2025-06-09T20:58:39.957+05:30  INFO 31713 --- [OnlineCourseManagementSystem] [  restartedMain] .OnlineCourseManagementSystemApplication : Started OnlineCourseManagementSyst
Enter 1: Add Course
Enter 2: Show Course By:
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
5
1: Instructor
2:Category
3:Date Range
1
Enter Instructor Name:
Mr. Techno
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0 where c1_0.instructor=?
Course{courseId=1, title='BCA', description='3 year dergree', instructor='Mr. Techno', category='IT Field', start=2025-08-01, endDate=2028-09-10}
2025-06-09T20:58:59.254+05:30  INFO 31713 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2025-06-09T20:58:59.257+05:30  INFO 31713 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 - Shutdown initiated.
2025-06-09T20:58:59.263+05:30  INFO 31713 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource      : HikariPool-1 - Shutdown completed.

Process finished with exit code 0
```

b. by Category



```
    Isolation level: undefined/unknown
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2025-06-09T20:59:53.701+05:30  INFO 32039 --- [OnlineCourseManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform availab
2025-06-09T20:59:53.750+05:30  INFO 32039 --- [OnlineCourseManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
2025-06-09T20:59:54.276+05:30  INFO 32039 --- [OnlineCourseManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on po
2025-06-09T20:59:54.298+05:30  INFO 32039 --- [OnlineCourseManagementSystem] [  restartedMain] .OnlineCourseManagementSystemApplication : Started OnlineCourseManagementSyst
Enter 1: Add Course
Enter 2: Show Course By:
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
5
1: Instructor
2:Category
3:Date Range
2
Enter Category:
IT Field
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0 where c1_0.category=?
Course{courseId=1, title='BCA', description='3 year dergree', instructor='Mr. Techno', category='IT Field', start=2025-08-01, endDate=2028-09-10}
2025-06-09T21:00:12.864+05:30  INFO 32039 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2025-06-09T21:00:12.865+05:30  INFO 32039 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated.
2025-06-09T21:00:12.871+05:30  INFO 32039 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.

Process finished with exit code 0
```

c. By Schedule Date Range



```
    Maximum pool size: undefined/unknown
2025-06-09T21:00:39.907+05:30  INFO 32306 --- [OnlineCourseManagementSystem] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform availab
2025-06-09T21:00:39.939+05:30  INFO 32306 --- [OnlineCourseManagementSystem] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFacto
2025-06-09T21:00:40.285+05:30  INFO 32306 --- [OnlineCourseManagementSystem] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on po
2025-06-09T21:00:40.303+05:30  INFO 32306 --- [OnlineCourseManagementSystem] [  restartedMain] .OnlineCourseManagementSystemApplication : Started OnlineCourseManagementSyst
Enter 1: Add Course
Enter 2: Show Course By:
Enter 3: Update Course Details
Enter 4: Delete course
Enter 5: Fetch Courses By:
5
1: Instructor
2:Category
3:Date Range
3
Enter Start Date (yyyy-mm-dd):
2020-05-10
Enter End Date (yyyy-mm-dd):
2029-05-10
Hibernate: select c1_0.course_id,c1_0.category,c1_0.description,c1_0.end_date,c1_0.instructor,c1_0.start_date,c1_0.title from courses c1_0 where c1_0.start_date>=? and c1_0.
Course{courseId=1, title='BCA', description='3 year dergree', instructor='Mr. Techno', category='IT Field', start=2025-08-01, endDate=2028-09-10}
2025-06-09T21:00:58.385+05:30  INFO 32306 --- [OnlineCourseManagementSystem] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2025-06-09T21:00:58.387+05:30  INFO 32306 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated.
2025-06-09T21:00:58.394+05:30  INFO 32306 --- [OnlineCourseManagementSystem] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.

Process finished with exit code 0
```