**Question 1:**

**1. Entity Class**

import javax.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

}

**2. Repository Interface**

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

public interface BookRepository extends JpaRepository<Book, Long> {

List<Book> findByAuthor(String author);

}

**3. Service or Controller Usage**

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public List<Book> getBooksByAuthor(String author) {

return bookRepository.findByAuthor(author);

}

public void addBook(Book book) {

bookRepository.save(book);

}

}

**Question 2:**

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **What it is** | A specification (interface) for ORM in Java | An implementation of JPA | A Spring module built on top of JPA (often using Hibernate) |
| **Type** | Interface / standard | Framework / tool | Abstraction / automation layer |
| **Who provides it** | Oracle (Java EE / Jakarta EE) | Red Hat | Spring Framework |
| **Purpose** | Defines standard APIs for ORM | Provides actual ORM logic | Simplifies JPA usage with minimal boilerplate |
| **Example** | @Entity, @Id, EntityManager | Session, Criteria, Lazy/Eager fetching | JpaRepository, @Query, CrudRepository |
| **SQL Requirement** | Yes, unless using JPQL or Criteria API | Yes, typically | Rarely — supports method name-based query generation |