**1. Demonstrate Implementation of Query Methods Feature of Spring Data JPA**

Spring Data JPA provides the Query Methods feature that allows you to define queries just by naming conventions in repository interfaces.

Example Entity: Country

@Entity

public class Country {

@Id

private String code;

private String name;

private int population;

private LocalDate independenceDate;

}

Repository with Query Methods:

public interface CountryRepository extends JpaRepository<Country, String> {

// Search containing text (case-insensitive)

List<Country> findByNameContainingIgnoreCase(String name);

// Filter with starting text

List<Country> findByNameStartingWith(String prefix);

// Greater than population

List<Country> findByPopulationGreaterThan(int population);

// Between two dates

List<Country> findByIndependenceDateBetween(LocalDate start, LocalDate end);

// Top N results

List<Country> findTop3ByOrderByPopulationDesc();

}

Sample Usage in Service/Test:

countryRepository.findByNameContainingIgnoreCase("ind");

countryRepository.findByNameStartingWith("U");

countryRepository.findByPopulationGreaterThan(1000000);

countryRepository.findByIndependenceDateBetween(LocalDate.of(1945, 1, 1), LocalDate.of(2000, 1, 1));

countryRepository.findTop3ByOrderByPopulationDesc();

**2. Demonstrate Implementation of O/R Mapping**

ORM mapping in JPA involves managing relationships using annotations such as:

* @ManyToOne, @OneToMany, @ManyToMany
* @JoinColumn, @JoinTable
* FetchType.LAZY, FetchType.EAGER
* mappedBy

🔹 Example: Country and State Relationship (One-to-Many, Many-to-One)

Country.java

CODE:

@Entity

public class Country {

@Id

private String code;

private String name;

@OneToMany(mappedBy = "country", fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private List<State> states = new ArrayList<>();

}

State.java

CODE:

@Entity

public class State {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "country\_code")

private Country country;

}

🔹 Example: Student and Course (Many-to-Many)

✅ Student.java

CODE:

@Entity

public class Student {

@Id

private Long id;

private String name;

@ManyToMany

@JoinTable(

name = "student\_course",

joinColumns = @JoinColumn(name = "student\_id"),

inverseJoinColumns = @JoinColumn(name = "course\_id"))

private List<Course> courses;

}

✅ Course.java

CODE:

@Entity

public class Course {

@Id

private Long id;

private String title;

@ManyToMany(mappedBy = "courses")

private List<Student> students;

}