6. Demonstrate implementation of DML using Spring Data JPA on a single database table

1. Setup and Configuration
   a. Maven Dependencies (pom.xml)

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
</dependencies>
```

   b. Application Properties

```
# H2 DB configuration
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
# Hibernate configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# Enable Hibernate logs
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql=TRACE
```

2. Define the Entity Class

```
import jakarta.persistence.*;
@Entity
@Table(name = "users")
public class User {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
```

```
 @Column(nullable = false)
 private String name;

 @Column(unique = true, nullable = false)
 private String email;

 // Getters and Setters
}
```

3. Define the Repository Interface

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

public interface UserRepository extends JpaRepository<User, Long> {
 // Custom Query Methods
 List<User> findByName(String name);
 User findByEmail(String email);
}
```

4. Service Class with DML Operations

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

import java.util.Optional;

@Service
public class UserService {

 @Autowired
 private UserRepository userRepository;

 // INSERT / UPDATE
 public User saveUser(User user) {
  return userRepository.save(user);
 }

 // SELECT by ID
 public Optional<User> getUserById(Long id) {
  return userRepository.findById(id);
 }

 // SELECT by Name
```

```
    public List<User> getUsersByName(String name) {
      return userRepository.findByName(name);
    }

    // DELETE
    public void deleteUserById(Long id) {
      userRepository.deleteById(id);
    }
  }
```

5. Sample Runner for Testing

```
import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class DemoRunner implements CommandLineRunner {

  @Autowired

  private UserService userService;

  @Override

  public void run(String... args) {

    // Insert

    User user = new User();

    user.setName("Alice");

    user.setEmail("alice@example.com");

    userService.saveUser(user);

    // Fetch by ID

    userService.getUserById(user.getId()).ifPresent(System.out::println);

    // Fetch by Name

    userService.getUsersByName("Alice").forEach(System.out::println);

    // Delete

    userService.deleteUserById(user.getId());

  }

}
```