● **Assignment title: Perform database operations using JPA Repository.**
● **Student Name: Omkar Deshpande**
● **Roll Number: A-26**
● **Submission Date:**

1. Introduction
Spring Data JPA is part of the larger Spring Data family that makes it easy to implement JPA-based repositories. It reduces the amount of boilerplate code required to implement data access layers.
JPA Repository is an interface provided by Spring Data JPA that provides CRUD (Create, Read, Update, Delete) operations and pagination support out of the box. It eliminates the need to write basic SQL queries manually.
Why JPA Repository is used:

- Reduces boilerplate code
- Provides built-in CRUD operations
- Supports custom query methods
- Easy pagination and sorting
- Type-safe repository implementation

2. Tools and Technologies Used

- **Java**: Version 17
- **Spring Boot**: Version 3.0.0
- **Spring Data JPA**: For database operations
- **MySQL**: Relational database management system
- **Maven**: Build tool and dependency management
- **IDE**: IntelliJ IDEA

Project Structure :

```
product-database-jpa/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── example/
│   │   │           └── productdemo/
│   │   │               ├── ProductDemoApplication.java
│   │   │               ├── entity/
│   │   │               │   └── Product.java
│   │   │               ├── repository/
│   │   │               │   └── ProductRepository.java
│   │   │               ├── service/
│   │   │               │   └── ProductService.java
│   │   │               └── controller/
│   │   │                   └── ProductController.java
│   │   └── resources/
│   │       ├── application.properties
│   │       └── data.sql
│   └── test/
│       └── java/
│           └── com/
│               └── example/
│                   └── productdemo/
│                       └── ProductDemoApplicationTests.java
```

```
├── pom.xml
└── README.md
```

1. ProductDemoApplication.java

```
package com.example.productdemo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ProductDemoApplication {

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

2. ProductRepository.java

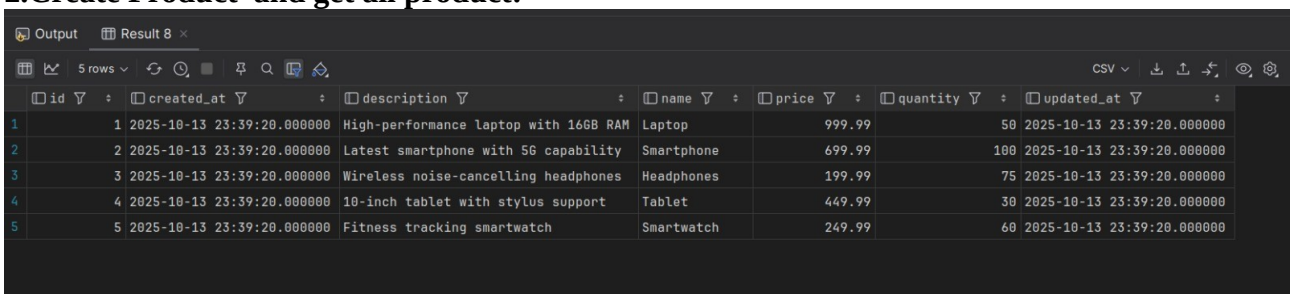
```
package com.example.productdemo.repository;

import com.example.productdemo.entity.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.math.BigDecimal;
import java.util.List;
import java.util.Optional;

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {

    // Find products by name (custom query method)
    List<Product> findByNameContainingIgnoreCase(String name);

    // Find products by price range
    List<Product> findByPriceBetween(BigDecimal minPrice, BigDecimal maxPrice);

    // Find products with quantity greater than specified value
    List<Product> findByQuantityGreaterThan(Integer quantity);

    // Custom query using JPQL
    @Query("SELECT p FROM Product p WHERE p.name LIKE %:keyword% OR p.description
LIKE %:keyword%")
    List<Product> searchProducts(@Param("keyword") String keyword);

    // Find product by name (exact match)
    Optional<Product> findByName(String name);
```

}

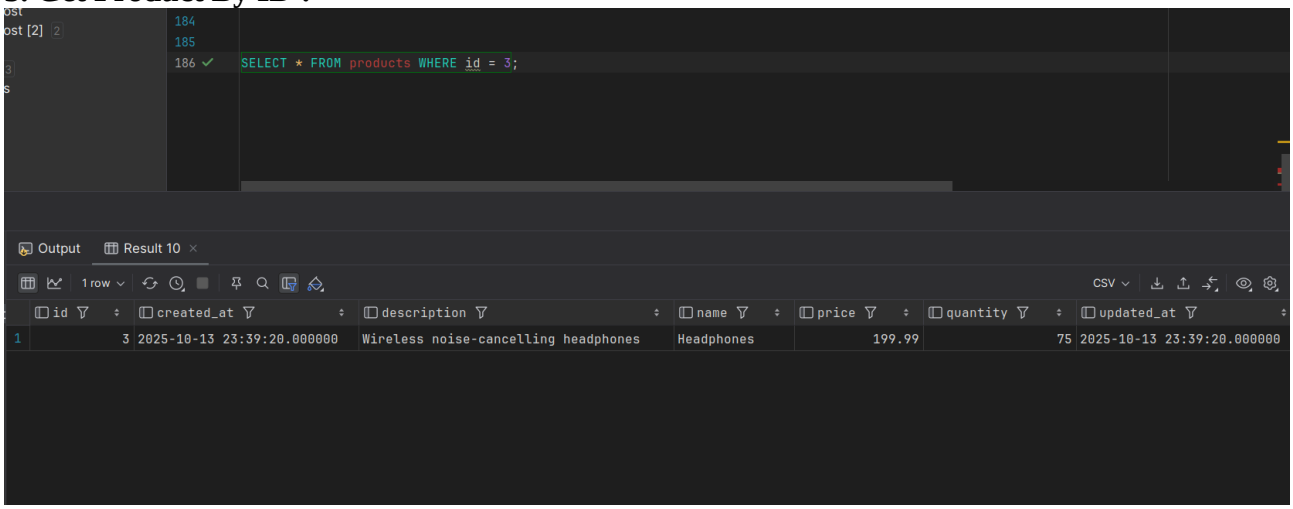
Output :

1 . Before Operations :

2.Create Product and get all product:



3. Get Product By ID :



4.Update Product :