

- **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);

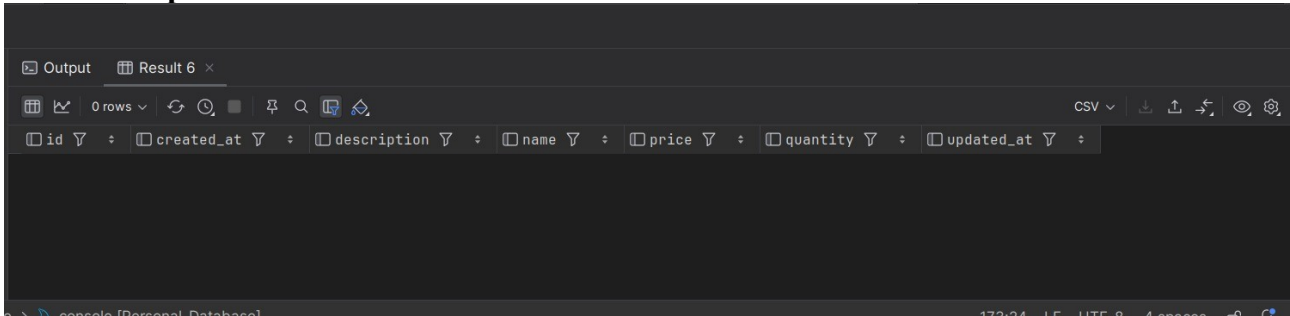
    // Custom query using JPQL
    @Query("SELECT p FROM Product p WHERE p.name LIKE %:keyword% OR p.description LIKE %:keyword%")
    List<Product> searchProducts(@Param("keyword") String keyword);

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

}

## Output :

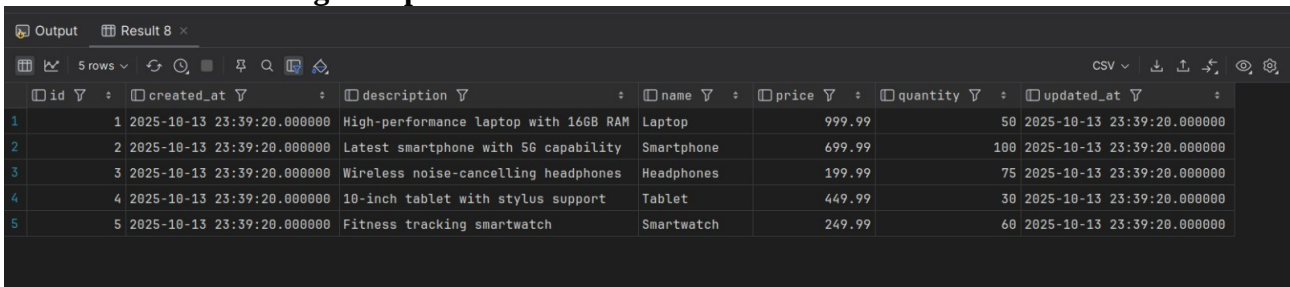
### 1 . Before Operations :



The screenshot shows a database client interface with a table named 'Result 6'. The table has 0 rows. The columns are: id, created\_at, description, name, price, quantity, and updated\_at.

id	created_at	description	name	price	quantity	updated_at
----	------------	-------------	------	-------	----------	------------

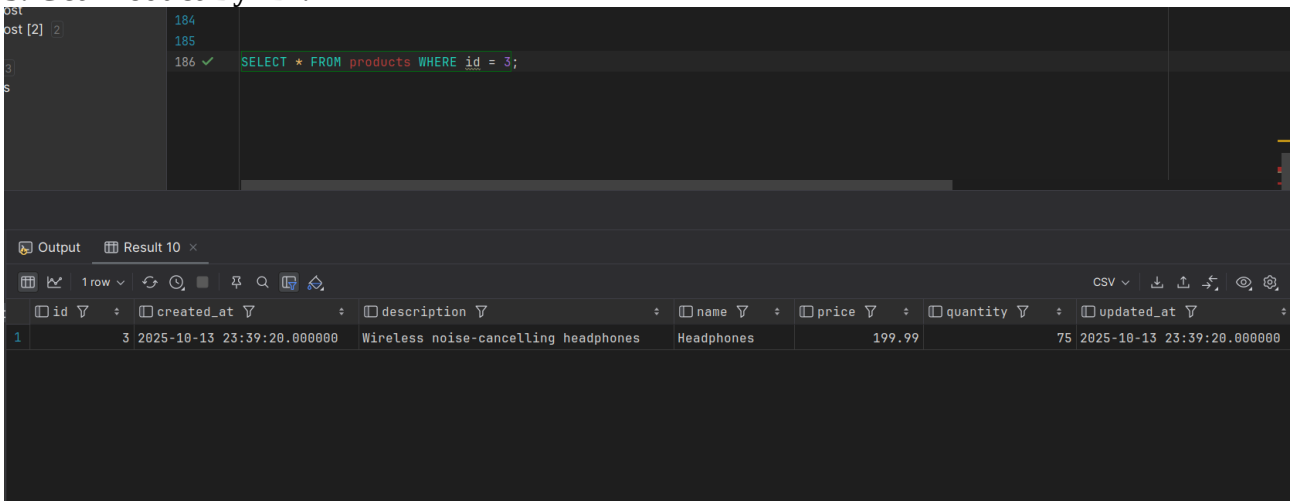
### 2.Create Product and get all product:



The screenshot shows a database client interface with a table named 'Result 8'. The table has 5 rows. The columns are: id, created\_at, description, name, price, quantity, and updated\_at.

id	created_at	description	name	price	quantity	updated_at
1	2025-10-13 23:39:20.000000	High-performance laptop with 16GB RAM	Laptop	999.99	50	2025-10-13 23:39:20.000000
2	2025-10-13 23:39:20.000000	Latest smartphone with 5G capability	Smartphone	699.99	100	2025-10-13 23:39:20.000000
3	2025-10-13 23:39:20.000000	Wireless noise-cancelling headphones	Headphones	199.99	75	2025-10-13 23:39:20.000000
4	2025-10-13 23:39:20.000000	10-inch tablet with stylus support	Tablet	449.99	30	2025-10-13 23:39:20.000000
5	2025-10-13 23:39:20.000000	Fitness tracking smartwatch	Smartwatch	249.99	60	2025-10-13 23:39:20.000000

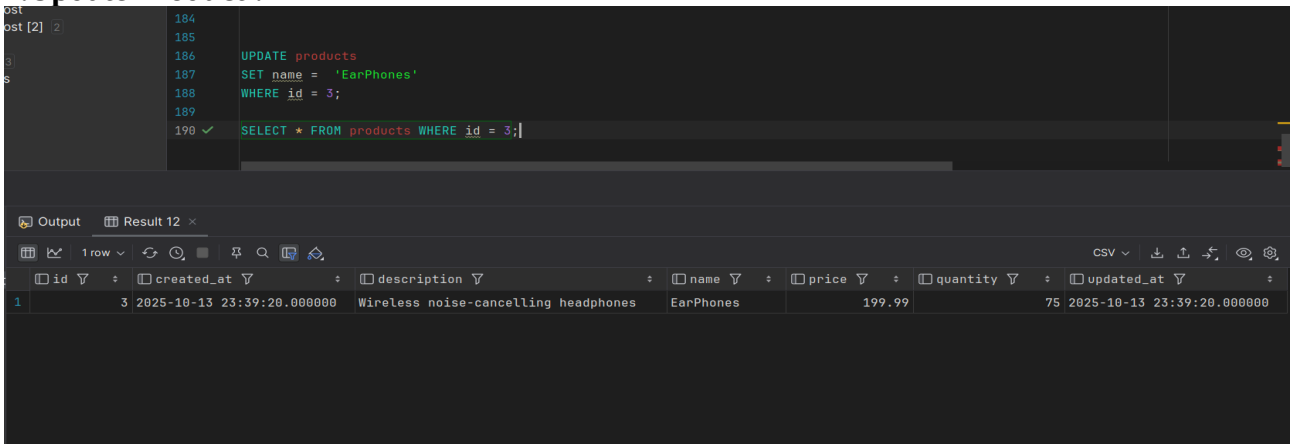
### 3. Get Product By ID :



The screenshot shows a database client interface with a SQL query and its result. The query is: `SELECT * FROM products WHERE id = 3;`. The result is a single row with the following data:

id	created_at	description	name	price	quantity	updated_at
3	2025-10-13 23:39:20.000000	Wireless noise-cancelling headphones	Headphones	199.99	75	2025-10-13 23:39:20.000000

### 4.Update Product :



The screenshot shows a database client interface with a SQL query and its result. The query is: `UPDATE products SET name = 'EarPhones' WHERE id = 3;`. The result is a single row with the following data:

id	created_at	description	name	price	quantity	updated_at
3	2025-10-13 23:39:20.000000	Wireless noise-cancelling headphones	EarPhones	199.99	75	2025-10-13 23:39:20.000000