**Demonstrate writing Hibernate Query Language and Native Query**

Hibernate Query Language (HQL) and Java Persistence Query Language (JPQL) are object-oriented query languages used for querying data from databases using Java classes instead of direct table names. These queries are portable and can be used with any relational database.

HQL is specific to Hibernate, while JPQL is a standardized version defined by the JPA specification. Both are very similar in syntax and allow developers to perform database operations like select, update, delete, and use aggregate functions such as AVG(), COUNT(), and MAX().

In Spring Data JPA, custom HQL or JPQL queries can be written using the @Query annotation. For example, to select all products of a certain category using JPQL, you would write:

@Query("SELECT p FROM Product p WHERE p.category = ?1")

List<Product> findByCategory(String category);

Native SQL queries can also be written using the @Query annotation by setting the nativeQuery attribute to true. This allows direct use of database-specific SQL syntax when necessary.

Fetch strategies like JOIN FETCH are used in HQL to eagerly load associated entities in a single query, which helps improve performance and prevent the N+1 select problem.

**EXAMPLE CODE:**

package com.example.demo.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private double price;

private String category;

// Getters and setters

}

**ProductRepository.java**

package com.example.demo.repository;

import com.example.demo.entity.Product;

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

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

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface ProductRepository extends JpaRepository<Product, Long> {

@Query("SELECT p FROM Product p WHERE p.category = ?1")

List<Product> findByCategory(String category);

@Query("SELECT AVG(p.price) FROM Product p")

Double findAveragePrice();

@Query("SELECT COUNT(p) FROM Product p WHERE p.category = ?1")

Long countByCategory(String category);

@Query(value = "SELECT \* FROM product WHERE price > ?1", nativeQuery = true)

List<Product> findProductsWithPriceGreaterThan(double price);

@Query("SELECT p FROM Product p WHERE p.price > :minPrice")

List<Product> findByPriceGreaterThan(@Param("minPrice") double price);

}

**ProductController.java**

package com.example.demo.controller;

import com.example.demo.entity.Product;

import com.example.demo.repository.ProductRepository;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/products")

public class ProductController {

@Autowired

private ProductRepository repo;

@GetMapping("/category")

public List<Product> byCategory(@RequestParam String category) {

return repo.findByCategory(category);

}

@GetMapping("/average")

public Double averagePrice() {

return repo.findAveragePrice();

}

@GetMapping("/count")

public Long countCategory(@RequestParam String category) {

return repo.countByCategory(category);

}

@GetMapping("/native")

public List<Product> nativeQuery(@RequestParam double price) {

return repo.findProductsWithPriceGreaterThan(price);

}

@GetMapping("/greater")

public List<Product> greaterThan(@RequestParam double minPrice) {

return repo.findByPriceGreaterThan(minPrice);

}

}

**SAMPLE OUTPUT:**

[

{

"id": 1,

"name": "Samsung Galaxy",

"price": 45000,

"category": "Mobile"

},

{

"id": 2,

"name": "iPhone 13",

"price": 75000,

"category": "Mobile"

}

]