**Product.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "product")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private Double price;

private Double ramSize;

private Double cpuSpeed;

private Double weight;

private String os;

private String cpu;

private Double hardDiskSize;

private Integer customerRating;

// Getters & Setters

@Override

public String toString() {

return "Product [id=" + id + ", name=" + name + ", price=" + price + ", ramSize=" + ramSize +

", cpuSpeed=" + cpuSpeed + ", weight=" + weight + ", os=" + os + ", cpu=" + cpu +

", hardDiskSize=" + hardDiskSize + ", customerRating=" + customerRating + "]";

}

}

**ProductRepository.java – Interface**

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Product;

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

public interface ProductRepository extends JpaRepository<Product, Long> {

// No methods needed for Criteria Query

}

**ProductService.java**

package com.example.ormlearn.service;

import com.example.ormlearn.model.Product;

import jakarta.persistence.\*;

import jakarta.persistence.criteria.\*;

import org.springframework.stereotype.Service;

import java.util.\*;

@Service

public class ProductService {

@PersistenceContext

private EntityManager entityManager;

public List<Product> searchProducts(Double minRam, String os, Double minCpuSpeed, Integer minRating) {

CriteriaBuilder cb = entityManager.getCriteriaBuilder();

CriteriaQuery<Product> query = cb.createQuery(Product.class);

Root<Product> root = query.from(Product.class);

List<Predicate> predicates = new ArrayList<>();

if (minRam != null) {

predicates.add(cb.greaterThanOrEqualTo(root.get("ramSize"), minRam));

}

if (os != null && !os.isEmpty()) {

predicates.add(cb.equal(root.get("os"), os));

}

if (minCpuSpeed != null) {

predicates.add(cb.greaterThanOrEqualTo(root.get("cpuSpeed"), minCpuSpeed));

}

if (minRating != null) {

predicates.add(cb.greaterThanOrEqualTo(root.get("customerRating"), minRating));

}

query.select(root).where(cb.and(predicates.toArray(new Predicate[0])));

return entityManager.createQuery(query).getResultList();

}

}

**OrmLearnApplication.java**

package com.example.ormlearn;

import com.example.ormlearn.model.Product;

import com.example.ormlearn.service.ProductService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.\*;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static ProductService productService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

productService = context.getBean(ProductService.class);

testCriteriaQuery();

}

public static void testCriteriaQuery() {

LOGGER.info("Start Criteria Query");

List<Product> products = productService.searchProducts(8.0, "Windows", 2.5, 4);

products.forEach(p -> LOGGER.debug("Product: {}", p));

LOGGER.info("End Criteria Query");

}

}