**EXERCISE 10**

*HIBERNATE – SPECIFIC ANNOTATIONS*

package com.example.EmployeeManagementSystem.model;

import org.hibernate.annotations.BatchSize;

import org.hibernate.annotations.Formula;

import org.hibernate.annotations.Type;

import javax.persistence.\*;

import java.util.Date;

@Entity

@Table(name = "employees")

@BatchSize(size = 10)

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name", nullable = false)

private String name;

@Column(name = "email", unique = true, nullable = false)

private String email;

@Type(type = "org.hibernate.type.NumericBooleanType")

@Column(name = "is\_active", nullable = false)

private Boolean isActive;

@Formula("(select count(\*) from projects p where p.employee\_id = id)")

private int projectCount;

@Temporal(TemporalType.DATE)

@Column(name = "hire\_date")

private Date hireDate;

// Getters and Setters

}

#Implement Collection Mapping with @collectionId

package com.example.EmployeeManagementSystem.model;

import org.hibernate.annotations.CollectionId;

import org.hibernate.annotations.GenericGenerator;

import org.hibernate.annotations.Type;

import javax.persistence.\*;

import java.util.ArrayList;

import java.util.Collection;

@Entity

@Table(name = "departments")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name", nullable = false)

private String name;

@ElementCollection

@CollectionTable(name = "department\_locations", joinColumns = @JoinColumn(name = "department\_id"))

@GenericGenerator(name = "sequence-gen", strategy = "sequence")

@CollectionId(columns = @Column(name = "location\_id"), generator = "sequence-gen", type = @Type(type = "long"))

private Collection<String> locations = new ArrayList<>();

// Getters and Setters

}

*CONFIGURE HIBERNATE DIALECT*

# application.properties

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

spring.jpa.properties.hibernate.jdbc.batch\_size=20

spring.jpa.properties.hibernate.order\_inserts=true

spring.jpa.properties.hibernate.order\_updates=true

spring.jpa.properties.hibernate.cache.use\_second\_level\_cache=true

spring.jpa.properties.hibernate.cache.region.factory\_class=org.hibernate.cache.jcache.JCacheRegionFactory

spring.jpa.properties.hibernate.cache.use\_query\_cache=true

*IMPLEMENTING BATCH PROCESSING*

package com.example.EmployeeManagementSystem.service;

import com.example.EmployeeManagementSystem.model.Employee;

import com.example.EmployeeManagementSystem.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void saveEmployeesInBatch(List<Employee> employees) {

int batchSize = 20;

for (int i = 0; i < employees.size(); i++) {

employeeRepository.save(employees.get(i));

if (i % batchSize == 0 && i > 0) {

// Flush and clear the persistence context every batchSize entities

employeeRepository.flush();

employeeRepository.clear();

}

}

}

}