**EXERCISE-10:**

**Hibernate Specific Annotations:**

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

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

@Column(name = "description")

private String description;

}

**For class Employee,**

@Entity

@DynamicUpdate

public class Employee {

}

**Usage of @SQLDelete and @where:**

@Entity

@SQLDelete(sql = "UPDATE employee SET deleted = true WHERE id = ?")

@Where(clause = "deleted = false")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private boolean deleted = false;

}

**application.properties:**

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

spring.jpa.properties.hibernate.jdbc.batch\_size=50

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

spring.jpa.properties.hibernate.cache.region.factory\_class=org.hibernate.cache.ehcache.EhCacheRegionFactory

spring.jpa.show-sql=true

**application.yml:**

spring:

jpa:

properties:

hibernate:

dialect: org.hibernate.dialect.MySQLDialect

jdbc:

batch\_size: 50

cache:

use\_second\_level\_cache: true

region:

factory\_class: org.hibernate.cache.ehcache.EhCacheRegionFactory

show-sql: true

**Batch Processing:**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void batchInsertEmployees(List<Employee> employees) {

int batchSize = 50; // Adjust batch size as needed

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

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

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

// Flush and clear the session to prevent memory issues

employeeRepository.flush();

employeeRepository.clear();

}

}

}

}