Session: Spring Data JPA with Hibernate Part 2

Assignment

Instructions for JPQL and Native SQL Query

Create an employeeTable table with the following fields: empId, empFirstName, empLastName, empSalary, empAge.

```sql
use newdb;

CREATE TABLE employeeTable(
empId int not null primary key auto_increment,
empFirstName varchar(255),
empLastName varchar(255),
empSalary int,
empAge int
);

select * from employeeTable;

insert into employeeTable values(101, 'Steven', 'King', 50000, 32);
insert into employeeTable(empFirstName, empLastName, empSalary, empAge) values('John', 'Wick', 2500000, 32);
insert into employeeTable(empFirstName, empLastName, empSalary, empAge) values('Lucky', 'Singh', 36000, 36);
```

```sql
11 • select * from employeeTable;
```

Result Grid | Filter Rows: | Edit:

empId	empFirstName	empLastName	empSalary	empAge
101	Steven	King	50000	32
102	John	Wick	25000	32
103	Lucky	Singh	36000	36
104	Lex	De Haan	17000	42
105	Alexander	Hunold	9000	22
106	Bruce	Ernst	8000	25
107	Paramveer	Singh	72000	27
108	David	Austin	28000	38
109	Diana	Lorentz	50000	28
110	Nancy	Greenberg	24000	27
111	Daniel	Faviet	12000	29
112	Shanta	Vollman	16000	34
113	Julia	Nayer	18000	42
114	Shushant	Singh	40000	30
115	Hazel	Khan	82000	31
NULL	NULL	NULL	NULL	NULL

Create an Employee entity having following fields: id, firstName, lastName, salary, age which maps to the table columns given in above.

```
@Entity
@Table(name = "employeeTable")
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "empId")
    private int id;
    @Column(name = "empFirstName")
    private String firstName;
    @Column(name = "empLastName")
    private String lastName;
    @Column(name = "empSalary")
    private int salary;
    @Column(name = "empAge")
    private int age;
```

JPQL

Display the first name, last name of all employees having salary greater than average salary ordered in ascending by their age and in descending by their salary.

```
@Query("   SELECT firstName, lastName\n" +
        "    FROM Employee\n" +
        "    WHERE salary >(SELECT AVG(salary) FROM Employee)\n" +
        "    ORDER BY salary DESC, age")
List<Object[]> findEmployeeOrderBySalaryAndAge();
```

Output:

```
FirstName  LastName
John        Wick
```

Update salary of all employees by a salary passed as a parameter whose existing salary is less than the average salary.

```
@Query("select id from Employee where salary <(select AVG(salary) from Employee) ")
List<Integer> getEmployeeIDWithSalLessThanAVG();
@Modifying
@Transactional
@Query("update Employee set salary = :updatedSalary where id = :givenId")
void updateEmpSalary(@Param("updatedSalary") int updatedSalary, @Param("givenId") int id);
```

Before Updating

```
11 •  select * from employeeTable;
```

Result Grid | Filter Rows: | Edit:

empId	empFirstName	empLastName	empSalary	empAge
101	Steven	King	50000	32
102	John	Wick	25000	32
103	Lucky	Singh	36000	36
104	Lex	De Haan	17000	42
105	Alexander	Hunold	9000	22
106	Bruce	Ernst	8000	25
107	Paramveer	Singh	72000	27
108	David	Austin	28000	38
109	Diana	Lorentz	50000	28
110	Nancy	Greenberg	24000	27
111	Daniel	Faviet	12000	29
112	Shanta	Vollman	16000	34
113	Julia	Nayer	18000	42
114	Shushant	Singh	40000	30
115	Hazel	Khan	82000	31
NULL	NULL	NULL	NULL	NULL

After Updating

```
select * from employeeTable;
```

Result Grid | Filter Rows: | Edit:

empId	empFirstName	empLastName	empSalary	empAge
101	Steven	King	50000	32
102	John	Wick	35000	32
103	Lucky	Singh	36000	36
104	Lex	De Haan	35000	42
105	Alexander	Hunold	35000	22
106	Bruce	Ernst	35000	25
107	Paramveer	Singh	72000	27
108	David	Austin	35000	38
109	Diana	Lorentz	50000	28
110	Nancy	Greenberg	35000	27
111	Daniel	Faviet	35000	29
112	Shanta	Vollman	35000	34
113	Julia	Nayer	35000	42
114	Shushant	Singh	40000	30
115	Hazel	Khan	82000	31
NULL	NULL	NULL	NULL	NULL

Delete all employees with minimum salary.

```
@Query("select min(salary) from Employee")
Integer minSalary();
@Modifying
@Transactional
@Query(value = "delete from Employee where salary = :minsalary")
void deleteMinSalary(@Param("minsalary")Integer minsalary);
```

Before Deleting

```
select * from employeeTable;
```

Result Grid | Filter Rows: | Edit:

empId	empFirstName	empLastName	empSalary	empAge
101	Steven	King	50000	32
102	John	Wick	35000	32
103	Lucky	Singh	36000	36
104	Lex	De Haan	35000	42
105	Alexander	Hunold	35000	22
106	Bruce	Ernst	35000	25
107	Paramveer	Singh	72000	27
108	David	Austin	35000	38
109	Diana	Lorentz	50000	28
110	Nancy	Greenberg	35000	27
111	Daniel	Faviet	35000	29
112	Shanta	Vollman	35000	34
113	Julia	Nayer	35000	42
114	Shushant	Singh	40000	30
115	Hazel	Khan	82000	31
NULL	NULL	NULL	NULL	NULL

After Deleting

```
select * from employeeTable;
```

Result Grid | Filter Rows: | Edit:

empId	empFirstName	empLastName	empSalary	empAge
101	Steven	King	50000	32
103	Lucky	Singh	36000	36
107	Paramveer	Singh	72000	27
109	Diana	Lorentz	50000	28
114	Shushant	Singh	40000	30
115	Hazel	Khan	82000	31
NULL	NULL	NULL	NULL	NULL

Native SQL Query

Display the id, first name, age of all employees where last name ends with "singh"

```
@Query(value="select empId, empFirstName, empAge from employeeTable" +
        " where empLastName Like '%singh'", nativeQuery = true)
List<Object[]>findAllEmployeeLikeNQ();
```

Output:

```
Hibernate: select empId, empFirstName, empAge from employeeTable where empLastName Like '%singh'
Id    FirstName    Age
103      Lucky      36
107      Paramveer      27
114      Shushant      30
```

Delete all employees with age greater than 40(Should be passed as a parameter)

```
@Modifying
@Transactional
@Query(value="delete from employeeTable where empAge>:agelimit", nativeQuery = true)
void deleteByAgeNQ(@Param("agelimit")int agelimit);
```

Output: Employees with id: 104, 113 are deleted from the database

```
select * from employeeTable;
```

Result Grid | Filter Rows: | Edit:

empId	empFirstName	empLastName	empSalary	empAge
101	Steven	King	50000	32
102	John	Wick	2500000	32
103	Lucky	Singh	36000	36
105	Alexander	Hunold	9000	22
106	Bruce	Ernst	8000	25
107	Paramveer	Singh	72000	27
108	David	Austin	28000	38
109	Diana	Lorentz	50000	28
110	Nancy	Greenberg	24000	27
111	Daniel	Faviet	12000	29
112	Shanta	Vollman	16000	34
114	Shushant	Singh	40000	30
115	Hazel	Khan	82000	31
NULL	NULL	NULL	NULL	NULL

Inheritance Mapping

Implement and demonstrate Single Table strategy.

```
use newdb;

create table payment(
id int PRIMARY KEY,
pmode varchar(5),
amount int ,
cardnumber varchar(20),
checknumber varchar(20)
);

select * from payment;
```

```java
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "pmode", discriminatorType = DiscriminatorType.STRING)
public class Payment {

    @Id
    private int id;
    private int amount;
```

```java
@Entity
@DiscriminatorValue(value = "cc")

public class CreditCard extends Payment {

    private String cardnumber;
```

```java
@Entity
@DiscriminatorValue(value = "ch")
public class Check extends Payment{

    private String checknumber;
```

Result Grid | Filter Rows: | Edit:

id	pmode	amount	cardnumber	checknumber
501	cc	1500	cc123	NULL
502	cc	1700	cc124	NULL
503	cc	1800	cc12345	NULL
701	ch	1500	NULL	ch1002
702	ch	1700	NULL	ch1003
703	ch	1800	NULL	ch1005
NULL	NULL	NULL	NULL	NULL

Implement and demonstrate Joined strategy.

```sql
use newdb;
create table payment(
id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
amount decimal(8,3)
);

create table card(
id int ,
cardnumber varchar(20),
 FOREIGN KEY (id)
REFERENCES payment(id)
);

create table bankcheck(
id int ,
checknumber varchar(20),
FOREIGN KEY (id)
REFERENCES payment(id)
);
```

```java
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class Payment {

    @Id
    private int id;
    private int amount;
```

```java
@Entity
@Table(name = "card")
@PrimaryKeyJoinColumn(name = "id")
public class CreditCard extends Payment {

    private String cardnumber;
```

```java
@Entity
@Table(name = "bankcheck")
@PrimaryKeyJoinColumn(name = "id")
public class Check extends Payment{

    private String checknumber;
```

```
22
23 •  select * from payment;
```

Result Grid | Filter Rows:

id	amount
501	1500.000
502	1700.000
503	1800.000
701	1500.000
702	1700.000
703	1800.000
NULL	NULL

```
30 •  select * from card;
```

Result Grid | Filter Rows:

id	cardnumber
501	cc123
502	cc124
503	cc12345

```
33
34 •  select * from bankcheck;
```

Result Grid | Filter Rows:

id	checknumber
701	ch1002
702	ch1003
703	ch1005

Implement and demonstrate Table Per Class strategy.

```
create table card(
id int PRIMARY KEY,
amount int,
cardnumber varchar(20)
);

create table bankcheck(
id int PRIMARY KEY,
amount int,
checknumber varchar(20)
);
```

```
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public class Payment {

    @Id
    private int id;
    private int amount;
```

```
@Entity
@Table(name = "card")
public class CreditCard extends Payment {

    private String cardnumber;
```

```
@Entity
@Table(name = "bankcheck")
public class Check extends Payment{

    private String checknumber;
```

```
select * from card;
```

Result Grid Filter Rows:

id	amount	cardnumber
501	1500	cc123
502	1700	cc124
503	1800	cc12345
NULL	NULL	NULL

```
select * from bankcheck;
```

Result Grid Filter Rows:

id	amount	checknumber
701	1500	ch1002
702	1700	ch1003
703	1800	ch1005
NULL	NULL	NULL

Component Mapping

Implement and demonstrate Embedded mapping using employee table having following fields: id, firstName, lastName, age, basicSalary, bonusSalary, taxAmount, specialAllowanceSalary.

```java
@Entity
@Table(name = "emp")
public class EmployeeComponentMapping {

    @Id
    private int id;
    private String firstName;
    private String lastName;
    private int age;

    @Embedded
    private SalaryComponentMapping salaryComponentMapping;
```

```java
@Embeddable
public class SalaryComponentMapping {

    private int basicSalary;
    private int bonusSalary;
    private int taxAmount;
    private int specialAllowanceSalary;
```

```java
@Service
public class EmployeeCMDaoService {

    @Autowired
    EmployeeComponentMappingRepository employeeComponentMappingRepository;

    public void createEmpCM()
    {
        EmployeeComponentMapping emp  = new EmployeeComponentMapping();
        emp.setId(301);
        emp.setFirstName("Raj");
        emp.setLastName("Verma");
        emp.setAge(27);
        SalaryComponentMapping sal = new SalaryComponentMapping();
        sal.setBasicSalary(25000);
        sal.setBonusSalary(5000);
        sal.setTaxAmount(2100);
        sal.setSpecialAllowanceSalary(5000);
        emp.setSalaryComponentMapping(sal);
        employeeComponentMappingRepository.save(emp);
    }
}
```

```sql
create table emp(
id int PRIMARY KEY,
firstName varchar(20),
lastName varchar(20),
age int ,
basicSalary int,
bonusSalary int,
taxAmount int,
specialAllowanceSalary int
);

select * from emp;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	firstName	lastName	age	basicSalary	bonusSalary	taxAmount	specialAllowanceSalary
301	Raj	Verma	27	25000	5000	2100	5000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL