# Session: Spring Data JPA with Hibernate Part 2 <u>Assignment</u>

## Instructions for JPQL and Native SQL Query

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

```
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);
```

1	11 • select * from employeeTable;								
<									
Result Grid 1 N Filter Rows: Edit:									
	empId	empFirstName	empLastName	empSalary	empAge				
<b>•</b>	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	HULL	NULL	NULL	NULL				

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

#### **JPOL**

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.

FirstName LastName John Wick

Output:

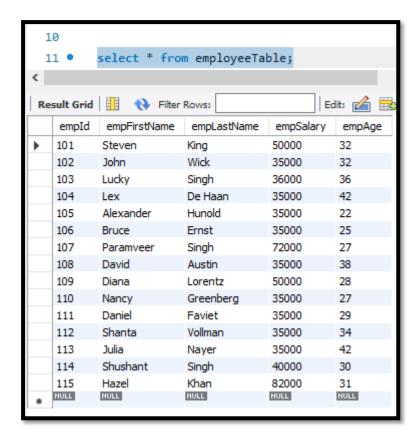
Update salary of all employees by a salary passed as a parameter whose existing salary is less than the average salary.

```
@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

1	11 • select * from employeeTable;								
<									
Re	Result Grid								
	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				

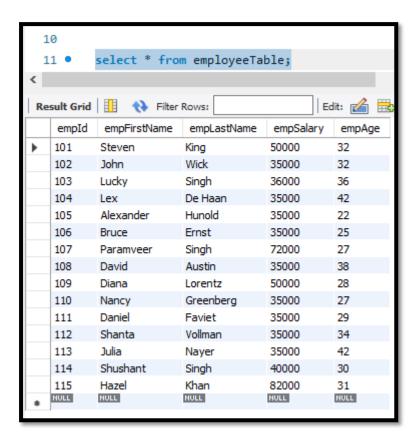
## After Updating



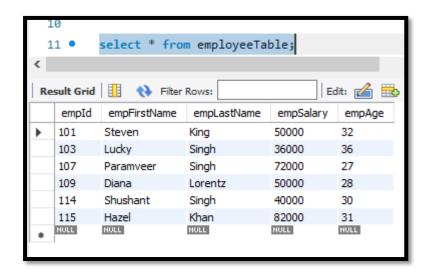
Delete all employees with minimum salary.

```
@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



## After Deleting



#### Native SQL Query

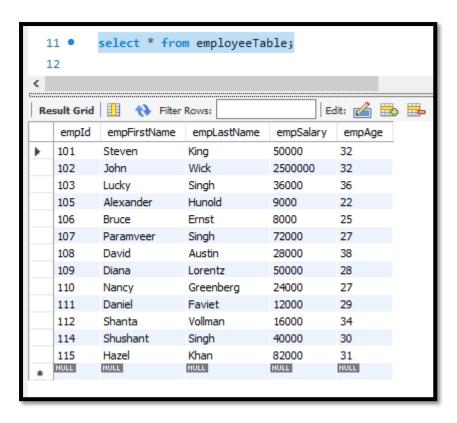
Display the id, first name, age of all employees where last name ends with "singh"

#### Output:

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



## Inheritance Mapping

Implement and demonstrate Single Table strategy.

```
use newdb;

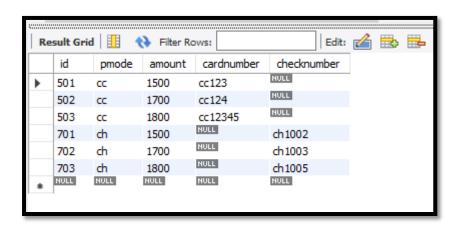
create table payment(
id int PRIMARY KEY,
pmode varchar(5),
amount int ,
cardnumber varchar(20),
checknumber varchar(20)
);

select * from payment;
```

```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@ODiscriminatorColumn(name = "pmode", discriminatorType = DiscriminatorType.STRING)
public class Payment {
    @Id
    private int id;
    private int amount;
```

```
@Entity
@DiscriminatorValue(value = "cc")
public class CreditCard extends Payment {
    private String cardnumber;
```

```
@Entity
@DiscriminatorValue(value = "ch")
public class Check extends Payment{
    private String checknumber;
```



Implement and demonstrate Joined strategy.

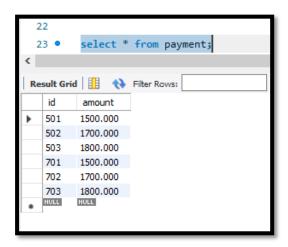
```
use newdb;
create table payment(
  id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  amount decimal(8,3)
  );
create table card(
  id int ,
  cardnumber varchar(20),
  FOREIGN KEY (id)
  REFERENCES payment(id)
  );
create table bankcheck(
  id int ,
  checknumber varchar(20),
  FOREIGN KEY (id)
  REFERENCES payment(id)
  reperson of the payment of
```

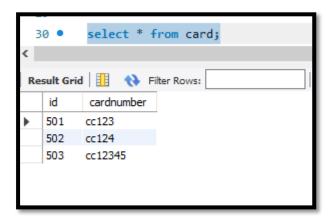
```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class Payment {

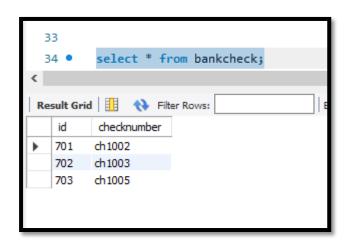
    @Id
    private int id;
    private int amount;
```

```
@CEntity
@Table(name = "card")
@@PrimaryKeyJoinColumn(name = "id")
public class CreditCard extends Payment {
    private String cardnumber;
```

```
@Entity
@Table(name = "bankcheck")
@@PrimaryKeyJoinColumn(name = "id")
public class Check extends Payment{
    private String checknumber;
```







Implement and demonstrate Table Per Class strategy.

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

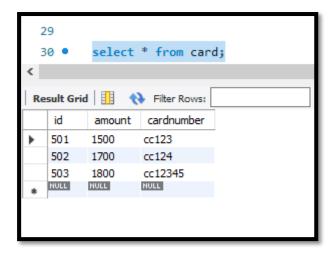
create table bankcheck(
  id int PRIMARY KEY,
  amount int,
  checknumber varchar(20)
);
```

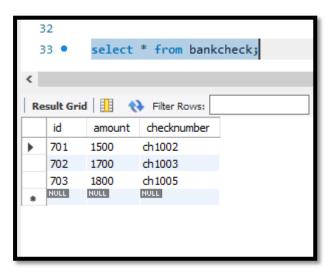
```
@CEntity
@CInheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public class Payment {

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

```
@Entity
@CTable(name = "card")
public class CreditCard extends Payment {
    private String cardnumber;
```

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





## **Component Mapping**

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

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

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

    @Embedded
    private SalaryComponentMapping salaryComponentMapping;
```

```
@Embeddable
public class SalaryComponentMapping {
    private int basicSalary;
    private int bonusSalary;
    private int taxAmount;
    private int specialAllowanceSalary;
```

```
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);
    }
```

