Session : Spring Data JPA with Hibernate Part 2

- **Instructions for JPQL and Native SQL Query**
  - **Create an employeeTable table with the following fields: empId, empFirstName, empLastName, empSalary, empAge.**
  - **Create an Employee entity having following fields: id, firstName, lastName, salary, age which maps to the table columns given in above.**

Employee.java

```
package com.example.employee1.entity;
import javax.persistence.*;
@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 double salary;
    @Column(name = "empAge")
    private int age;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
```

```java
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "Employee{" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", salary=" + salary +
                ", age=" + age +
                '}';
    }
}
```

```java
package com.example.employee1.repository;
import com.example.employee1.entity.Employee;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
public interface EmployeeRepository extends
CrudRepository<Employee,Integer> {
```

Result Grid Filter Rows: Export: Wrap Cell Content:

#	Field	Type	Null	Key	Default	Extra
1	empId	int(11)	NO	PRI	NULL	auto_increment
2	empFirstName	varchar(30)	YES		NULL	
3	empLastName	varchar(30)	YES		NULL	
4	empSalary	double	YES		NULL	
5	empAge	int(11)	YES		NULL	

Result Grid Filter Rows: Edit: Export

#	empId	empFirstName	empLastName	empSalary	empAge
1	3	tony	stark	40000	30
2	4	bruce	wayne	40000	31
3	5	karan	singh	45000	32
4	6	taranjeet	singh	45000	33
5	7	puginder	singh	50000	58
*	NULL	NULL	NULL	NULL	NULL

JPQL:

1. **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 avg(salary) from Employee")
public Double avgSalary();

@Test// average salary
public void testAvgSalary(){
   Double avgSalary=employeeRepository.avgSalary();
   System.out.println("----------------");
   System.out.println("average salary---->"+avgSalary);
   System.out.println("----------------");
}
```

Run: EmployeeApplicationTests.testAvgSalary

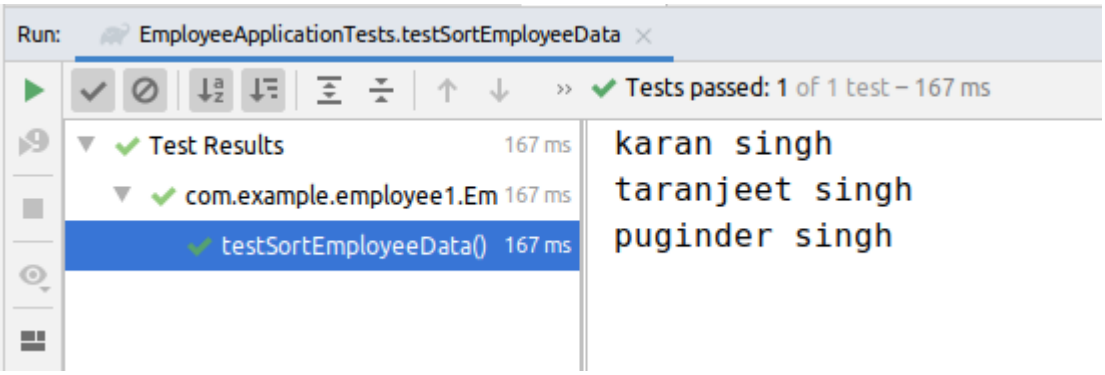
Tests passed: 1 of 1 test – 197 ms

Test Results 197 ms
com.example.employee1.Em 197 ms
testAvgSalary() 197 ms

```
----------------
average salary---->44000.0
----------------
```

```
    //jpql
    //Q-1
@Query("select firstName,lastName from Employee where
salary>(select avg(salary) from Employee) order by age asc, salary
desc")
public List<Object[]> sortEmployeeData();
```

```
//JPQL

@Test //Q-1 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.
void testSortEmployeeData(){
   List<Object[]> records=employeeRepository.sortEmployeeData();
   for(Object[] record:records){
      for (Object data:record){
         System.out.print(data+" ");
      }
      System.out.println("");
   }
}
```



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

```
//Q-2
@Modifying
@Query("update Employee set salary=:varSal where salary <:avgSal
")
public void updateEmployeeSalary(@Param("varSal") Double
varSal,@Param("avgSal") Double avgSal);

@Test //Q-2 Update salary of all employees by a salary passed as a
parameter
     // whose existing salary is less than the average salary.
@Transactional
@Rollback(false)
void testUpdateEmployeeSalary(){

employeeRepository.updateEmployeeSalary(30000.00,employeeRepositor
y.avgSalary());
   System.out.println("-----------------");
```

```
    System.out.println("salary updated");
    System.out.println("------------------");
}
```

Run: EmployeeApplicationTests.testAvgSalary

Tests passed: 1 of 1 test – 197 ms

Test Results 197 ms
com.example.employee1.Em 197 ms
testAvgSalary() 197 ms

```
---------------
average salary---->44000.0
----------------
```

Result Grid Filter Rows: Edit:

#	empId	empFirstName	empLastName	empSalary	empAge
1	3	tony	stark	40000	30
2	4	bruce	wayne	40000	31
3	5	karan	singh	45000	32
4	6	taranjeet	singh	45000	33
5	7	puginder	singh	50000	58

Test Results 219 ms
com.example.employee1.Em 219 ms
testUpdateEmployeeSala 219 ms

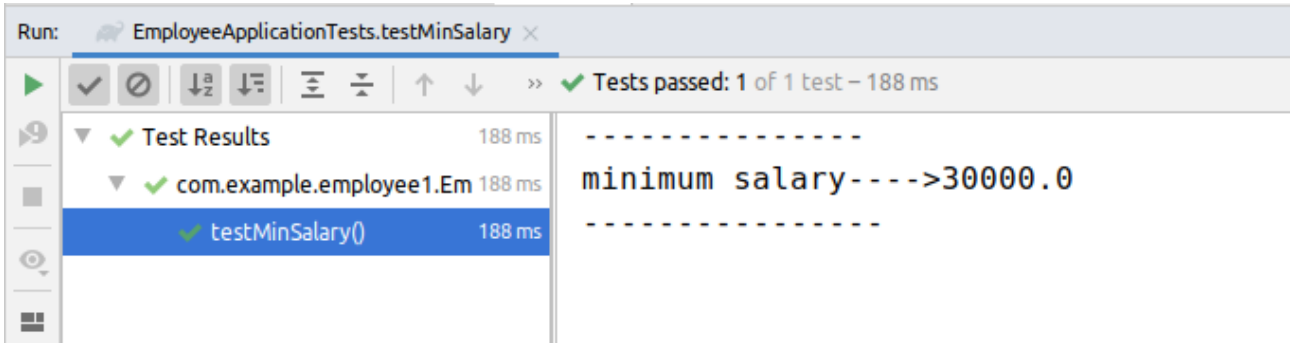
```
2020-04-10 02:37:24.104  IN
-----------------
salary updated
------------------
2020-04-10 02:37:24.300  IN
```

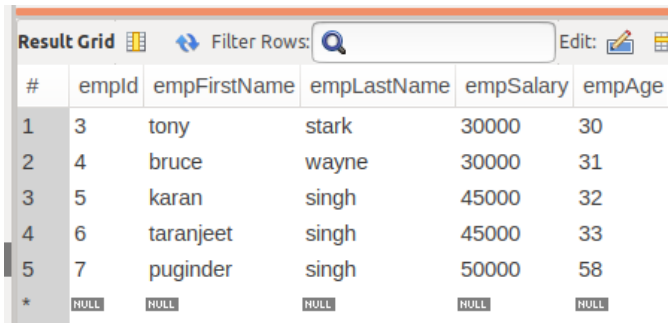
Result Grid Filter Rows: Edit:

#	empId	empFirstName	empLastName	empSalary	empAge
1	3	tony	stark	30000	30
2	4	bruce	wayne	30000	31
3	5	karan	singh	45000	32
4	6	taranjeet	singh	45000	33
5	7	puginder	singh	50000	58
*	NULL	NULL	NULL	NULL	NULL

3. **Delete all employees with minimum salary.**

```
    //Q-3
@Modifying
@Query("delete from Employee where salary=:minSal")
public void deleteEmployeeWithMinSal(@Param("minSal") Double
minSal);



@Test// min salary

public void testMinSalary(){
   Double minSalary=employeeRepository.minSalary();
   System.out.println("---------------");
   System.out.println("minimum salary---->"+minSalary);
   System.out.println("----------------");
}



@Test // Q-3 Delete all employees with minimum salary.
@Transactional
@Rollback(false)
public void testDeleteEmployeeWithMinSal(){

employeeRepository.deleteEmployeeWithMinSal(employeeRepository.min
Salary());
   System.out.println("-----------------");
   System.out.println("minimum salary deleted");
   System.out.println("------------------");
}
```





Result Grid | Filter Rows: | Edit:

#	empId	empFirstName	empLastName	empSalary	empAge
1	3	tony	stark	30000	30
2	4	bruce	wayne	30000	31
3	5	karan	singh	45000	32
4	6	taranjeet	singh	45000	33
5	7	puginder	singh	50000	58
*	NULL	NULL	NULL	NULL	NULL

Result Grid Filter Rows: Edit: Exp

#	empId	empFirstName	empLastName	empSalary	empAge
1	5	karan	singh	45000	32
2	6	taranjeet	singh	45000	33
3	7	puginder	singh	50000	58
*	NULL	NULL	NULL	NULL	NULL

Native SQL Query:

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

```
    // native SQl
    //Q-1
    @Query(value =" select empId,empFirstName,empAge from
employeeTable where empLastName like 'singh' ",nativeQuery = true)
    public List<Object[]> displayEmployeeDataNQ();


// Native Sql Querry
//Q-1 //Display the id, first name, age of all employees
// where last name ends with "singh"
@Test
public void testdisplayEmployeeDataNQ(){
   List<Object[]>
records=employeeRepository.displayEmployeeDataNQ();
   for (Object[] record:records) {
      for (Object data:record){
         System.out.print(data+" ");
      }
      System.out.println("");
   }
}
```

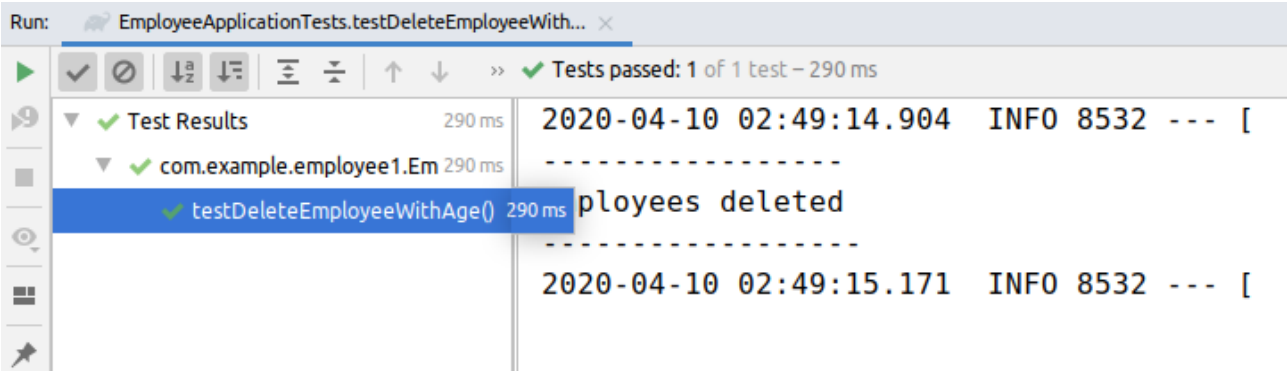
Tests passed: 1 of 1 test – 299 ms

Test Results 299 ms
com.example.employee1.Em 299 ms
testdisplayEmployeeData 299 ms

```
5 karan 32
6 taranjeet 33
7 puginder 58
```

2. **Delete all employees with age greater than 45(Should be passed as a parameter)**

```
 //Q-2
   @Modifying
   @Query(value = "delete from employeeTable where
empAge>:age",nativeQuery = true)
   void deleteEmployeeByAgeGreaterThan(@Param("age") int age);
}

//Q-2 Delete all employees with age greater than 45(Should be
passed as a parameter)

@Test
@Transactional
@Rollback(false)
public void testDeleteEmployeeWithAge(){
   employeeRepository.deleteEmployeeByAgeGreaterThan(45);
   System.out.println("-----------------");
   System.out.println("employees deleted");
   System.out.println("------------------");
```



Result Grid | Filter Rows: | Edit: | Exp

#	empId	empFirstName	empLastName	empSalary	empAge
1	5	karan	singh	45000	32
2	6	taranjeet	singh	45000	33
*	NULL	NULL	NULL	NULL	NULL

Inheritance Mapping:

1. **Implement and demonstrate Single Table strategy.**

Employee.java

```
package com.example.inheritmap.entity;
import javax.persistence.*;
@Entity
@Table(name = "employee_table")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "emp_type" ,discriminatorType
=DiscriminatorType.STRING)
public abstract 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 double salary;
    @Column(name = "empAge")
    private int age;
    public Employee(int id, String firstName, String lastName,
double salary, int age) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.salary = salary;
        this.age = age;
    }
    public Employee() {
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
```

```
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "Employee{" +
               "id=" + id +
               ", firstName='" + firstName + '\'' +
               ", lastName='" + lastName + '\'' +
               ", salary=" + salary +
               ", age=" + age +
               '}';
    }
}
```

RegularEmployee.java

```
package com.example.inheritmap.entity;
import com.example.inheritmap.entity.Employee;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
@Entity
@DiscriminatorValue("Regular Employee")
public class RegularEmployee extends Employee {
    private String project;
    public RegularEmployee(){
        super();
    }
    public RegularEmployee(int id, String firstName, String
lastName, double salary, int age,String project) {
        super(id, firstName, lastName, salary, age);
        this.project = project;
    }
    public String getProject() {
        return project;
    }
    @Override
    public String toString() {
```

```
        return "RegularEmployee{" +
                "id='" + this.getId() + '\'' +
                "name='" + this.getFirstName() +"
"+this.getLastName()+ '\'' +
                "age='" + this.getAge() + '\'' +
                "salary='" + this.getSalary() + '\'' +
                "project='" + project + '\'' +
                '}';
    }
}
```

TraineeEmployee.java

```
package com.example.inheritmap.entity;
import com.example.inheritmap.entity.Employee;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
@Entity
@DiscriminatorValue("Trainee Employee")
public class TraineeEmployee extends Employee {
    private String assessment;
    public TraineeEmployee(){
        super();
    }
    public TraineeEmployee(int id, String firstName, String
lastName, double salary, int age,String assessment) {
        super(id, firstName, lastName, salary, age);
        this.assessment = assessment;
    }
    public String getAssessment() {
        return assessment;
    }
}
```

8	8	steve	rogers	50000	35		NULL	NULL
9	12	gurbindar	singh	15000	36	Trainee Employee	NULL	drupal
10	13	taranjeet	singh	15000	36	Trainee Employee	NULL	dev-ops
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. **Implement and demonstrate Joined strategy.**

// payment.java

```
     package com.example.inheritmap3.entity;
import javax.persistence.*;
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Payment {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String payeeName;
    private double amount;
    public Payment(){
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public double getAmount() {
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
    public String getPayeeName() {
        return payeeName;
    }
    public void setPayeeName(String payeeName) {
        this.payeeName = payeeName;
    }
}
```

//credit-card.java

```
package com.example.inheritmap3.entity;

import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
@Entity
@PrimaryKeyJoinColumn(name = "id")
public class CreditCard extends Payment {
    private String cardNumber;
    public CreditCard(){
        super();
    }
    public String getCardNumber() {
        return cardNumber;
```

```
    }
    public void setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
    }
}
```

// cheque.java

```
package com.example.inheritmap3.entity;

import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
@Entity
@PrimaryKeyJoinColumn(name = "id")
public class Cheque extends Payment {
    private String chequeNumber;
    public Cheque(){
        super();
    }
    public String getChequeNumber() {
        return chequeNumber;
    }
    public void setChequeNumber(String chequeNumber) {
        this.chequeNumber = chequeNumber;
    }
}
```

```
1 •  SELECT * FROM employeeDB1.Payment;
```

Result Grid Filter Rows: Edit: Expor

#	id	amount	payeeName
1	17	13000	father
2	18	13000	father
3	19	13000	father
*	NULL	NULL	NULL

Result Grid Filter Rows:

#	cardNumber	id
1	cc-4321	18
2	cc-4321	19
*	NULL	NULL

Result Grid Filter Rows:

#	chequeNumber	id
1	cq-132418	20
*	NULL	NULL

3. **Implement and demonstrate Table Per Class strategy.**

// training.java

```
package com.example.inheritmap2.entity;
import javax.persistence.*;
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Training {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String traineeName;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTraineeName() {
        return traineeName;
    }
    public void setTraineeName(String traineeName) {
        this.traineeName = traineeName;
    }
}
```

// Gym.java

```
     package com.example.inheritmap2.entity;
import javax.persistence.Entity;
@Entity
public class Gym extends Training {
    private String gymLocation;
    private int hours;
    public String getGymLocation() {
        return gymLocation;
    }
    public void setGymLocation(String gymLocation) {
        this.gymLocation = gymLocation;
    }
    public int getHours() {
        return hours;
    }
    public void setHours(int hours) {
        this.hours = hours;
    }
}
```

// yoga.java

```
     package com.example.inheritmap2.entity;
import javax.persistence.Entity;
@Entity
public class Yoga extends Training {
    private String centerLocation;
    private int hours;
    public String getCenterLocation() {
        return centerLocation;
    }
    public void setCenterLocation(String centerLocation) {
        this.centerLocation = centerLocation;
    }
    public int getHours() {
        return hours;
    }
    public void setHours(int hours) {
        this.hours = hours;
    }
}
```

Result Grid Filter Rows: Edit:

#	id	traineeName	gymLocation	hours
1	10	bani j	delhi	2
*	NULL	NULL	NULL	NULL

Result Grid Filter Rows: Edit:

#	id	traineeName	centerLocation	hours
1	9	piku	pune	1
*	NULL	NULL	NULL	NULL

Component Mapping:

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

```
package com.example.componentmap.entity;
import javax.persistence.*;
@Entity
public class Worker {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String firstName;
    private String lastName;
    private int age;
    @Embedded
    private Salary salary;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
```

```java
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public Salary getSalary() {
        return salary;
    }
    public void setSalary(Salary salary) {
        this.salary = salary;
    }
}
```

```java
package com.example.componentmap.entity;
import javax.persistence.Embeddable;
import javax.persistence.Entity;
@Embeddable
public class Salary {
    private double basicSalary;
    private double bonusSalary;
    private double taxAmount;
    private double specialAllowanceSalary;
    public double getBasicSalary() {
        return basicSalary;
    }
    public void setBasicSalary(double basicSalary) {
        this.basicSalary = basicSalary;
    }
    public double getBonusSalary() {
        return bonusSalary;
    }
    public void setBonusSalary(double bonusSalary) {
        this.bonusSalary = bonusSalary;
    }
    public double getTaxAmount() {
        return taxAmount;
    }
    public void setTaxAmount(double taxAmount) {
        this.taxAmount = taxAmount;
    }
    public double getSpecialAllowanceSalary() {
        return specialAllowanceSalary;
```

```
    }
    public void setSpecialAllowanceSalary(double
specialAllowanceSalary) {
        this.specialAllowanceSalary = specialAllowanceSalary;
    }
}
```

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

#	id	age	firstName	lastName	basicSalary	bonusSalary	specialAllowanceSalary	taxAmount
1	11	34	tony	stark	13000	4000	500	2000
2	12	34	tony	stark	13000	4000	500	2000
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL