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.**

```
mysql> use SpringDataJPA2;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table employeetable(
    -> empid int PRIMARY KEY AUTO_INCREMENT,
    -> empfirstname varchar(20),
    -> emplastname varchar(20),
    -> empsalary double,
    -> empage int
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> desc employeetable;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| empid        | int(11)     | NO   | PRI | NULL    | auto_increment |
| empfirstname | varchar(20) | YES  |     | NULL    |                |
| emplastname  | varchar(20) | YES  |     | NULL    |                |
| empsalary    | double      | YES  |     | NULL    |                |
| empage       | int(11)     | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql>
```

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

CODE

```
package com.SpringData.SpringDataJPA2_Assignment2.entity;

import javax.persistence.*;

@Entity
@Table(name = "employeetable")
public class Employee {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "empid")
  private Integer id;
  @Column(name = "empfirstname")
  private String firstName;
  @Column(name = "emplastname")
  private String lastName;
  @Column(name = "empsalary")
  private Double salary;
  @Column(name = "empage")
  private Integer age;

  public Integer getId() {
     return id;
  }

  public void setId(Integer 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 Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Employee{" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", salary=" + salary +
                ", age=" + age +
                '}';
    }
}
```

```
mysql> use SpringDataJPA2;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_SpringDataJPA2 |
+--------------------------+
| employeetable            |
+--------------------------+
1 row in set (0.00 sec)

mysql> desc employeetable;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| empid        | int(11)     | NO   | PRI | NULL    | auto_increment |
| empfirstname | varchar(20) | YES  |     | NULL    |                |
| emplastname  | varchar(20) | YES  |     | NULL    |                |
| empsalary    | double      | YES  |     | NULL    |                |
| empage       | int(11)     | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql>
```

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.**

   <u>CODE</u>

<u>EmployeeRepository.java</u>

```
@Repository
public interface EmployeeRepository extends CrudRepository<Employee,Integer> {
  //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 from Employee " +
        "where salary > (select AVG(salary) from Employee) " +
        "Order By age ASC,salary DESC ")
  List<Object[]> findAllEmployeePartialData();
}
```

EmployeeService.java

```
//PARTIAL DATA
public List<Object[]> findAllEmployeePartial() {
  List<Object[]> employeeList = employeeRepository.findAllEmployeePartialData();
  return employeeList;
}
```

EmployeeController.java

```
@GetMapping("/find-all-partial")
public List<Object[]> findEmployeesPartialData()
{
  List<Object[]> employees = employeeService.findAllEmployeePartial();
  return employees;
}
```

OUTPUT

```
mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Mayanka      | Ahuja       |     95000 |     33 |
|     2 | Aayushi      | Thani       |     55000 |     25 |
|     3 | Simran       | Verma       |     35000 |     28 |
|     4 | Deepak       | Kumar       |     80000 |     55 |
|     5 | Shashi       | Sharma      |     60000 |     45 |
|     6 | Pragya       | Jain        |     45000 |     23 |
|     7 | Abhimanyu    | Mathur      |     70000 |     23 |
+-------+--------------+-------------+-----------+--------+
7 rows in set (0.00 sec)
```

GET http://localhost:8080/employees/find-all-partial

Pretty Raw Preview Visualize JSON


```
[
    [
        "Abhimanyu",
        "Mathur"
    ],
    [
        "Mayanka",
        "Ahuja"
    ],
    [
        "Deepak",
        "Kumar"
    ]
]
```


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

   CODE

EmployeeController.java

```
@GetMapping("/update/{increment}")
public String updateEmployeeSalary(@PathVariable Double increment)
{
  employeeService.updateEmployee(increment);
  return "Result Updated";
}
```

EmployeeService.java

```
//UPDATE THE EMPLOYEE SALARY
public void updateEmployee(Double increment){
      Iterable<Employee> list = employeeRepository.findAll();
      Iterator<Employee> iterator = list.iterator();
      Double sum =0d;
      int count =0;
      while (iterator.hasNext())
      {
         Employee e = iterator.next();
         sum = sum + e.getSalary();
         count++;
```

```
    }
    Double average = sum/count;
    employeeRepository.updateAllEmployeeSalary(increment,average);
}
```

EmployeeRepository.java

```
//Update salary of all employees by a salary passed as a parameter whose existing salary is less than the average salary.
@Modifying
@Transactional
@Query("update Employee SET salary=salary+:increment WHERE salary<:average")
void updateAllEmployeeSalary(@Param("increment") Double increment,@Param("average") Double average);
```

OUTPUT

```
mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Mayanka      | Ahuja       |     95000 |     33 |
|     2 | Aayushi      | Thani       |     55000 |     25 |
|     5 | Shashi       | Sharma      |     60000 |     45 |
|     6 | Pragya       | Jain        |     45000 |     23 |
|     7 | Abhimanyu    | Mathur      |     70000 |     23 |
+-------+--------------+-------------+-----------+--------+
5 rows in set (0.00 sec)

mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Mayanka      | Ahuja       |     95000 |     33 |
|     2 | Aayushi      | Thani       |     55500 |     25 |
|     5 | Shashi       | Sharma      |     60500 |     45 |
|     6 | Pragya       | Jain        |     45500 |     23 |
|     7 | Abhimanyu    | Mathur      |     70000 |     23 |
+-------+--------------+-------------+-----------+--------+
5 rows in set (0.00 sec)
```

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

CODE

EmployeeController.java

```
@GetMapping("/delete/{salary}")
public String deleteEmployee(@PathVariable Double salary){
  employeeService.deleteEmployee(salary);
  return "Employee With Minimum Salary Deleted";
}
```

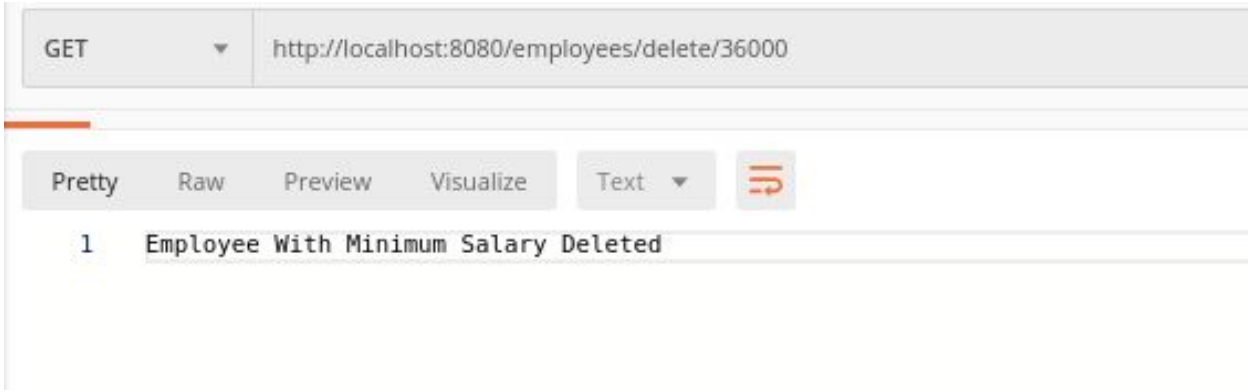
EmployeeService.java

```
//DELETE THE EMPLOYEE WITH MINIMUM SALARY
public void deleteEmployee(Double salary){
  employeeRepository.deleteAllEmployeesMinSalary(salary);
}
```

EmployeeRepository.java

```
//Delete all employees with minimum salary.
@Modifying
@Transactional
@Query("delete from Employee where salary<:minsalary")
void deleteAllEmployeesMinSalary(@Param("minsalary") Double minSalary);
```

OUTPUT



```
File  Edit  View  Search  Terminal  Help
mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Mayanka      | Ahuja       |     95000 |     33 |
|     2 | Aayushi      | Thani       |     55000 |     25 |
|     3 | Simran       | Verma       |     35000 |     28 |
|     4 | Deepak       | Kumar       |     80000 |     55 |
|     5 | Shashi       | Sharma      |     60000 |     45 |
|     6 | Pragya       | Jain        |     45000 |     23 |
|     7 | Abhimanyu    | Mathur      |     70000 |     23 |
|     8 | Rajiv        | singh       |     90000 |     57 |
+-------+--------------+-------------+-----------+--------+
8 rows in set (0.00 sec)

mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Mayanka      | Ahuja       |     95000 |     33 |
|     2 | Aayushi      | Thani       |     55000 |     25 |
|     4 | Deepak       | Kumar       |     80000 |     55 |
|     5 | Shashi       | Sharma      |     60000 |     45 |
|     6 | Pragya       | Jain        |     45000 |     23 |
|     7 | Abhimanyu    | Mathur      |     70000 |     23 |
|     8 | Rajiv        | singh       |     90000 |     57 |
+-------+--------------+-------------+-----------+--------+
7 rows in set (0.00 sec)
```

Native SQL Query:

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

   <u>CODE</u>

<u>EmployeeController.java</u>

```
@GetMapping("/last-name")
public List<Employee> findEmployeeLastNameNQ()
{
```

```
  List<Employee> employee = employeeService.findEmployeeLastName();
  return employee;
}
```

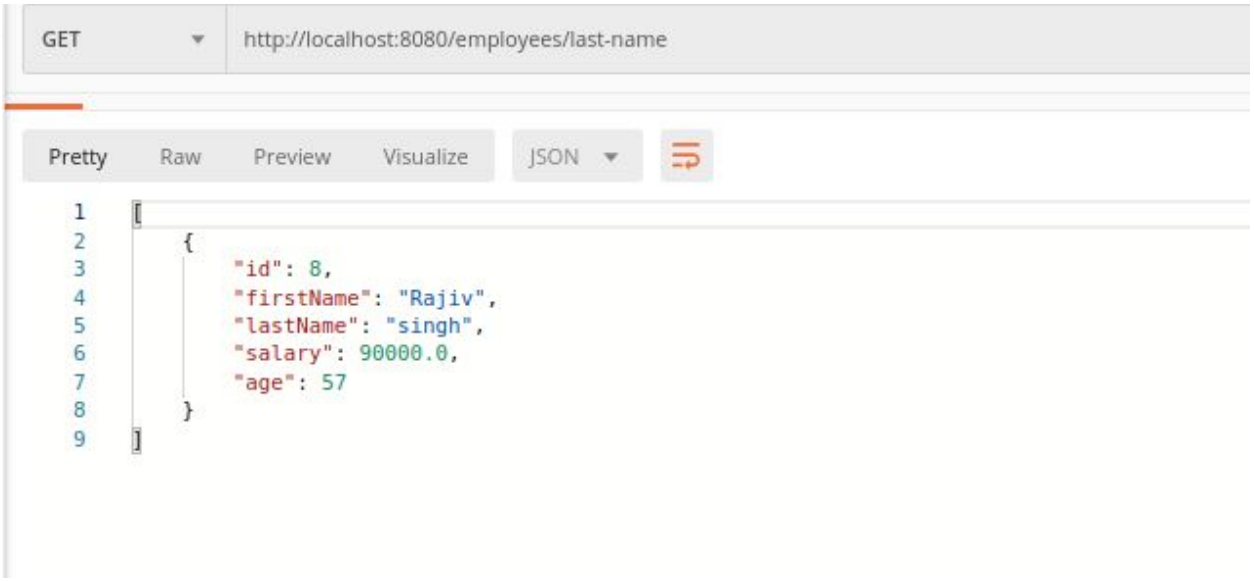
EmployeeService.java

```
//Native SQL Query - Display the id, first name, age of all employees where last name ends with "singh"
public List<Employee> findEmployeeLastName()
{
  List<Employee> lastNameNQ= employeeRepository.findEmployeeLastNameNQ("singh");
  return lastNameNQ;
}
```

EmployeeRepository.java

```
@Query(value = "select * from employeetable where emplastname=:lastname", nativeQuery = true)
List<Employee> findEmployeeLastNameNQ(@Param("lastname") String lastname);
```

OUTPUT



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

   CODE

EmployeeController.java

```
@GetMapping("/delete-greater-than-45")
public String deleteEmployeeGreaterThan45()
{
  employeeService.deleteEmployeeGreaterThan45();
  return "Deleted Employee Records Greater Than 45";
}
```

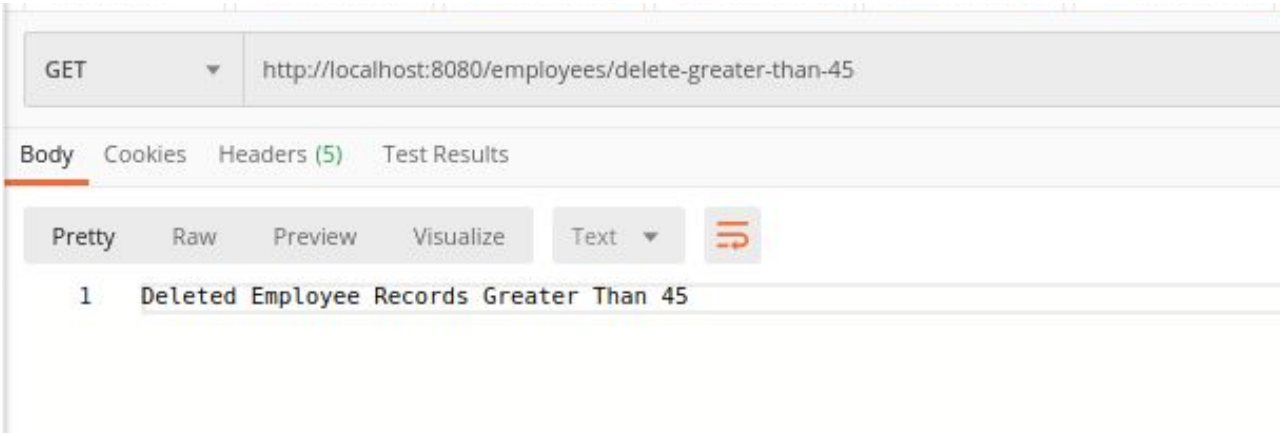
EmployeeService.java

```
//Native SQL Query - Delete all employees with age greater than 45(Should be passed as a parameter)
public void deleteEmployeeGreaterThan45()
{
  employeeRepository.deleteEmployeeGreaterThan45(45);
}
```

EmployeeRepository.java

```
//Delete all employees with age greater than 45(Should be passed as a parameter)
@Modifying
@Transactional
@Query(value = "delete from employeetable where empage>:age", nativeQuery = true)
void  deleteEmployeeGreaterThan45(@Param("age") Integer age);
```

OUTPUT



```
mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Mayanka      | Ahuja       |     95000 |     33 |
|     2 | Aayushi      | Thani       |     55000 |     25 |
|     4 | Deepak       | Kumar       |     80000 |     55 |
|     5 | Shashi       | Sharma      |     60000 |     45 |
|     6 | Pragya       | Jain        |     45000 |     23 |
|     7 | Abhimanyu    | Mathur      |     70000 |     23 |
|     8 | Rajiv        | singh       |     90000 |     57 |
+-------+--------------+-------------+-----------+--------+
7 rows in set (0.00 sec)

mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Mayanka      | Ahuja       |     95000 |     33 |
|     2 | Aayushi      | Thani       |     55000 |     25 |
|     5 | Shashi       | Sharma      |     60000 |     45 |
|     6 | Pragya       | Jain        |     45000 |     23 |
|     7 | Abhimanyu    | Mathur      |     70000 |     23 |
+-------+--------------+-------------+-----------+--------+
5 rows in set (0.00 sec)
```

Inheritance Mapping:

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

   <u>CODE</u>

<u>Payment.java</u>

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

  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;
    }
}
```

Check.java

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

    public String getChecknumber() {
        return checknumber;
    }

    public void setChecknumber(String checknumber) {
        this.checknumber = checknumber;
    }
}
```

CreditCard.java

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

    public String getCardnumber() {
        return cardnumber;
    }

    public void setCardnumber(String cardnumber) {
        this.cardnumber = cardnumber;
```

```
  }
}
```

PaymentController.java

```
@RestController
@RequestMapping("/payments")
public class PaymentController {

  @Autowired
  PaymentService paymentService;

  @PostMapping("/credit-card")
  public Payment createCreditCardPayment(@RequestBody CreditCard creditCard)
  {
     return paymentService.createPayment(creditCard);
  }

  @PostMapping("/check")
  public Payment createCheckPayment(@RequestBody Check check)
  {
     return paymentService.createCheckPayment(check);
  }

}
```

PaymentService.java

```
@Service
public class PaymentService {

  @Autowired
  PaymentRepository paymentRepository;

  public Payment createPayment(CreditCard creditCard)
  {
     return paymentRepository.save(creditCard);
  }

  public Payment createCheckPayment(Check check)
  {
     return paymentRepository.save(check);
  }
```

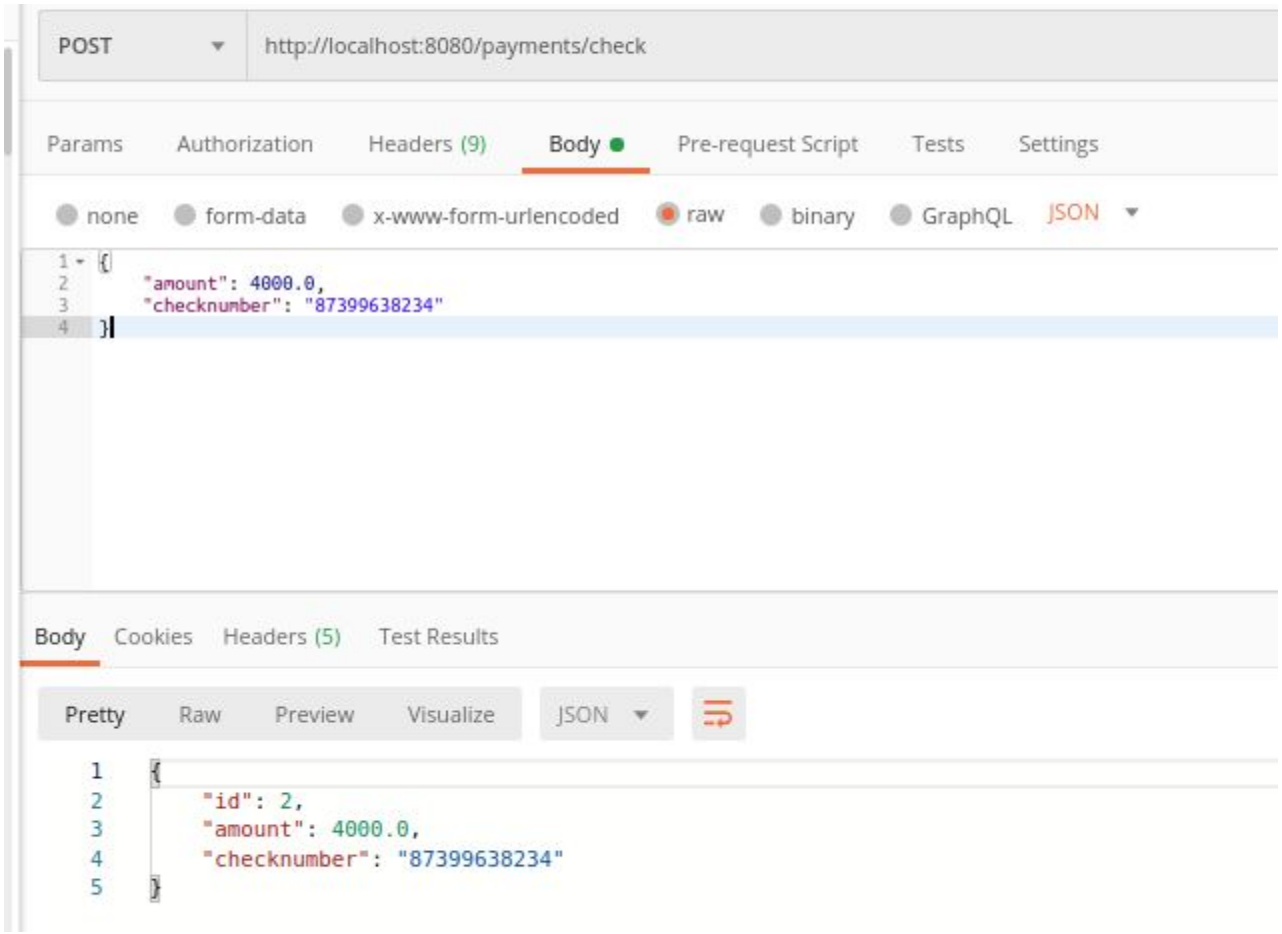
}

PaymentRepository.java

```
public interface PaymentRepository extends CrudRepository<Payment,Integer> {

}
```

OUTPUT



```
mysql> select * from payment;
+-------+----+--------+-------------+------------+
| pmode | id | amount | checknumber | cardnumber |
+-------+----+--------+-------------+------------+
| cc    |  1 |   2000 | NULL        | 99999234   |
| ch    |  2 |   4000 | 87399638234 | NULL       |
+-------+----+--------+-------------+------------+
2 rows in set (0.00 sec)
```

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

   CODE

Payment.java

```
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Payment {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private int id;
  private double amount;

  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;
  }
}
```

Check.java

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

  public String getChecknumber() {
     return checknumber;
  }

  public void setChecknumber(String checknumber) {
     this.checknumber = checknumber;
  }
}
```

CreditCard.java

```
@Entity
public class CreditCard extends Payment {
  private String cardnumber;

  public String getCardnumber() {
     return cardnumber;
  }

  public void setCardnumber(String cardnumber) {
     this.cardnumber = cardnumber;
  }
}
```

PaymentController.java

```
  @Autowired
  PaymentService paymentService;

  @PostMapping("/credit-card")
  public Payment createCreditCardPayment(@RequestBody CreditCard creditCard)
  {
     return paymentService.createPayment(creditCard);
  }
```

```
  @PostMapping("/check")
  public Payment createCheckPayment(@RequestBody Check check)
  {
     return paymentService.createCheckPayment(check);
  }
}
```

OUTPUT

```
mysql> show tables;
+--------------------------+
| Tables_in_SpringDataJPA2 |
+--------------------------+
| bankcheck                |
| credit_card              |
| employeetable            |
| hibernate_sequence       |
| payment                  |
+--------------------------+
5 rows in set (0.01 sec)

mysql> select * from hibernate_sequence;
+----------+
| next_val |
+----------+
|        1 |
+----------+
```

POST http://localhost:8080/payments/check

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "amount": 56000.0,
3     "checknumber": "AA8548483984"
4 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "id": 2,
3     "amount": 56000.0,
4     "checknumber": "AA8548483984"
5 }
```

```
mysql> select * from hibernate_sequence;
+----------+
| next_val |
+----------+
|        3 |
+----------+
1 row in set (0.01 sec)

mysql> select * from credit_card;
+----+--------+------------+
| id | amount | cardnumber |
+----+--------+------------+
|  1 |   4000 | 8887638234 |
+----+--------+------------+
1 row in set (0.00 sec)

mysql> select * from bankcheck;
+----+--------+--------------+
| id | amount | checknumber  |
+----+--------+--------------+
|  2 |  56000 | AA8548483984 |
+----+--------+--------------+
1 row in set (0.00 sec)

mysql>
```

3. **Implement and demonstrate Joined strategy.**

   <u>CODE</u>

<u>Payment.java</u>

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Payment {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private int id;
```

```
  private double amount;

  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;
  }
}
```

Check.java

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

  public String getChecknumber() {
      return checknumber;
  }

  public void setChecknumber(String checknumber) {
      this.checknumber = checknumber;
  }
}
```

CreditCard.java

```
@Entity
//@DiscriminatorValue("cc")
@PrimaryKeyJoinColumn(name = "id")
public class CreditCard extends Payment {
```

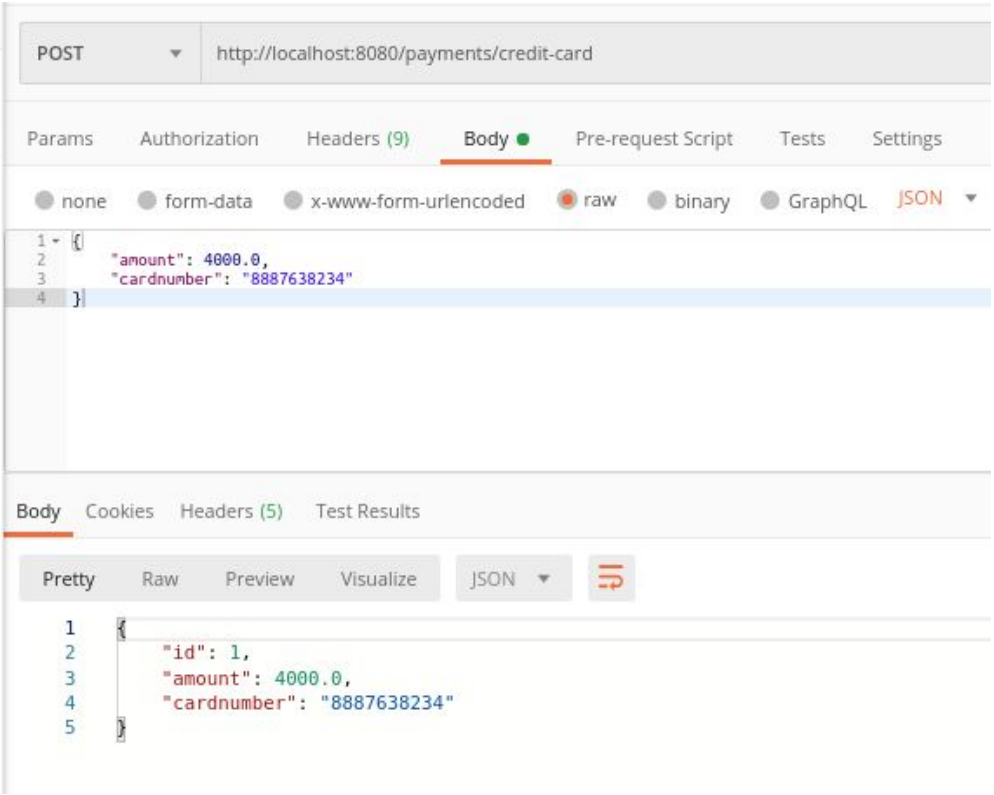
```
    private String cardnumber;

    public String getCardnumber() {
        return cardnumber;
    }

    public void setCardnumber(String cardnumber) {
        this.cardnumber = cardnumber;
    }
}
```

OUTPUT

```
mysql> show tables;
+---------------------------+
| Tables_in_SpringDataJPA2 |
+---------------------------+
| employeetable             |
+---------------------------+
1 row in set (0.00 sec)
```



POST http://localhost:8080/payments/check

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "amount": 56000.0,
3     "checknumber": "AA8548483984"
4 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "id": 2,
3     "amount": 56000.0,
4     "checknumber": "AA8548483984"
5 }
```

```
mysql> select * from payment;
+----+--------+
| id | amount |
+----+--------+
|  1 |   4000 |
|  2 |  56000 |
+----+--------+
2 rows in set (0.00 sec)

mysql> select * from credit_card;
+------------+----+
| cardnumber | id |
+------------+----+
| 8887638234 |  1 |
+------------+----+
1 row in set (0.00 sec)

mysql> select * from bankcheck;
+--------------+----+
| checknumber  | id |
+--------------+----+
| AA8548483984 |  2 |
+--------------+----+
1 row in set (0.00 sec)

mysql>
```

Component Mapping:

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

   <u>CODE</u>

Employee.java

```
@Entity
@Table(name = "employeemapping")
public class EmployeeMapping {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String firstname;
  private String lastname;
  private Integer age;
  @Embedded
  private Salary salary;

  public Integer getId() {
     return id;
  }

  public void setId(Integer 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 Integer getAge() {
     return age;
  }
```

```
  public void setAge(Integer age) {
      this.age = age;
  }

  public Salary getSalary() {
      return salary;
  }

  public void setSalary(Salary salary) {
      this.salary = salary;
  }
}
```

Salary.java

```
@Embeddable
public class Salary {

  private Integer basicsalary;
  private Integer bonussalary;
  private Integer taxamount;
  private Integer specialallowancesalary;

  public Integer getBasicsalary() {
      return basicsalary;
  }

  public void setBasicsalary(Integer basicsalary) {
      this.basicsalary = basicsalary;
  }

  public Integer getBonussalary() {
      return bonussalary;
  }

  public void setBonussalary(Integer bonussalary) {
      this.bonussalary = bonussalary;
  }

  public Integer getTaxamount() {
      return taxamount;
  }

  public void setTaxamount(Integer taxamount) {
```

```
        this.taxamount = taxamount;
    }

    public Integer getSpecialallowancesalary() {
        return specialallowancesalary;
    }

    public void setSpecialallowancesalary(Integer specialallowancesalary) {
        this.specialallowancesalary = specialallowancesalary;
    }
}
```

EmployeeController.java

```
@RestController
@RequestMapping("/employeesmapping")
public class EmployeeControllerMapping {

    @Autowired
    EmployeeServiceMapping employeeService;


    @PostMapping("/create")
    public String employeeCreate()
    {
        employeeService.employeeCreate();
        return "User Added";
    }

    @GetMapping("/find-all")
    public List<EmployeeMapping> findAllEmployee()
    {
        List<EmployeeMapping> employees = employeeService.findAllEmployee();
        return employees;
    }

}
```

EmployeeService.java

```
@Service
public class EmployeeServiceMapping {
```

```
  @Autowired
  EmployeeRepositoryMapping employeeRepositoryMapping;

  //CREATE A EMPLOYEE
  public void employeeCreate()
  {
     EmployeeMapping employeeMapping = new EmployeeMapping();
     employeeMapping.setFirstname("Aayushi");
     employeeMapping.setLastname("Thani");
     Salary salary = new Salary();
     salary.setBasicsalary(50000);
     salary.setBonussalary(6000);
     salary.setSpecialallowancesalary(3000);
     salary.setTaxamount(5000);
     employeeMapping.setSalary(salary);
     employeeRepositoryMapping.save(employeeMapping);
  }

  //FETCH THE LIST OF ALL EMPLOYEES
  public List<EmployeeMapping> findAllEmployee()
  {
     List<EmployeeMapping> employees = (List<EmployeeMapping>)
employeeRepositoryMapping.findAll();
     return employees;
  }
}
```

EmployeeRepository.java

```
@Repository
public interface EmployeeRepositoryMapping extends
CrudRepository<EmployeeMapping,Integer> {
}
```

OUTPUT

```
mysql> show tables;
+---------------------------+
| Tables_in_SpringDataJPA2 |
+---------------------------+
| bankcheck                 |
| credit_card               |
| employeetable             |
| payment                   |
+---------------------------+
4 rows in set (0.00 sec)

mysql> show tables;
+---------------------------+
| Tables_in_SpringDataJPA2 |
+---------------------------+
| bankcheck                 |
| credit_card               |
| employeemapping           |
| employeetable             |
| hibernate_sequence        |
| payment                   |
+---------------------------+
6 rows in set (0.00 sec)
```

```
mysql> desc employeemapping;
+------------------------+--------------+------+-----+---------+-------+
| Field                  | Type         | Null | Key | Default | Extra |
+------------------------+--------------+------+-----+---------+-------+
| id                     | int(11)      | NO   | PRI | NULL    |       |
| age                    | int(11)      | YES  |     | NULL    |       |
| firstname              | varchar(255) | YES  |     | NULL    |       |
| lastname               | varchar(255) | YES  |     | NULL    |       |
| basicsalary            | int(11)      | YES  |     | NULL    |       |
| bonussalary            | int(11)      | YES  |     | NULL    |       |
| specialallowancesalary | int(11)      | YES  |     | NULL    |       |
| taxamount              | int(11)      | YES  |     | NULL    |       |
+------------------------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Untitled Request

POST http://localhost:8080/employeesmapping/create

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1   User Added
```

```
mysql> select * from employeemapping;
+----+------+-----------+----------+-------------+-------------+-----------------------+-----------+
| id | age  | firstname | lastname | basicsalary | bonussalary | specialallowancesalary | taxamount |
+----+------+-----------+----------+-------------+-------------+-----------------------+-----------+
|  1 | NULL | Aayushi   | Thani    |       50000 |        6000 |                  3000 |      5000 |
+----+------+-----------+----------+-------------+-------------+-----------------------+-----------+
1 row in set (0.00 sec)
```

GET http://localhost:8080/employeesmapping/find-all

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests

Query Params

KEY	VALUE
Key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "id": 1,
        "firstname": "Aayushi",
        "lastname": "Thani",
        "age": null,
        "salary": {
            "basicsalary": 50000,
            "bonussalary": 6000,
            "taxamount": 5000,
            "specialallowancesalary": 3000
        }
    }
]
```