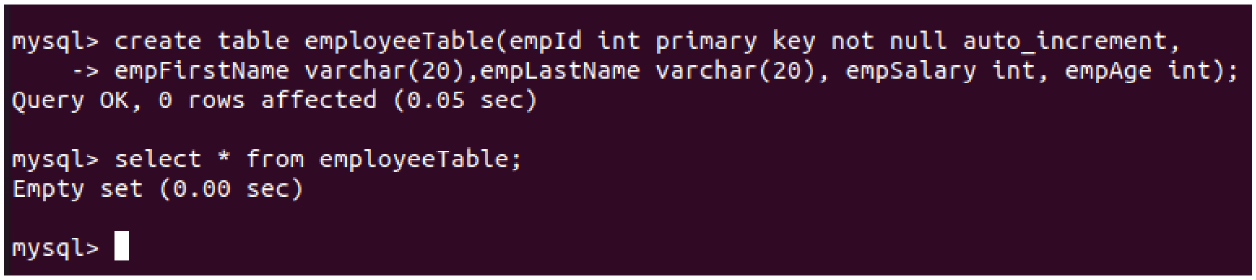
- **Instructions for JPQL and Native SQL Query**
  - **Create an employeeTable table with the following fields: empId, empFirstName, empLastName, empSalary, empAge.**
  - **Create an Employee entity having following fields: id, firstName, lastName, salary, age which maps to the table columns given in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



```
mysql> create table employeeTable(empId int primary key not null auto_increment,
    -> empFirstName varchar(20),empLastName varchar(20), empSalary int, empAge int);
Query OK, 0 rows affected (0.05 sec)

mysql> select * from employeeTable;
Empty set (0.00 sec)

mysql> 
```

<Employee.java>

```
package com.springdatajpawithhibernatepart2.Assignment.entities;

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 int salary;

  @Column(name = "empage")
  private int age;
```

```java
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 int getSalary() {
    return salary;
}

public void setSalary(int 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 +
            '}';
  }
}
```

JPQL:

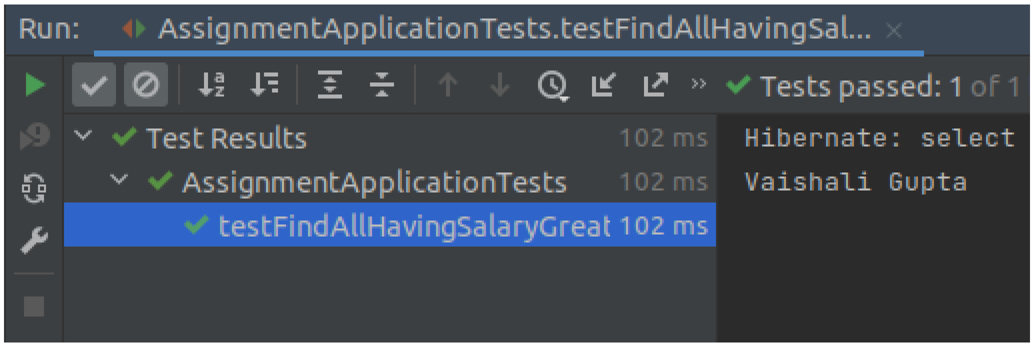
1. **Display the first name, last name of all employees having salary greater than average salary ordered in ascending by their age and in descending by their salary.**

<EmployeeRepository.java>

```
@Query("Select firstName,lastName from Employee where salary > " +
     "(select avg(salary) from Employee) order by age asc, salary desc" )
List<Object[]> findAllHavingSalaryGreaterThanAvgSalary();
```

<ApplicationTest.java>

```
@Test
public void testFindAllHavingSalaryGreaterThanAvgSalary(){
  List<Object[]> empData = employeeRepository.findAllHavingSalaryGreaterThanAvgSalary();
  for (Object[] emp : empData)
     System.out.println(emp[0] + " " + emp[1]);
}
```



```
Run: AssignmentApplicationTests.testFindAllHavingSal...
Tests passed: 1 of 1
Test Results                          102 ms
  AssignmentApplicationTests          102 ms
    testFindAllHavingSalaryGreat      102 ms

Hibernate: select
Vaishali Gupta
```

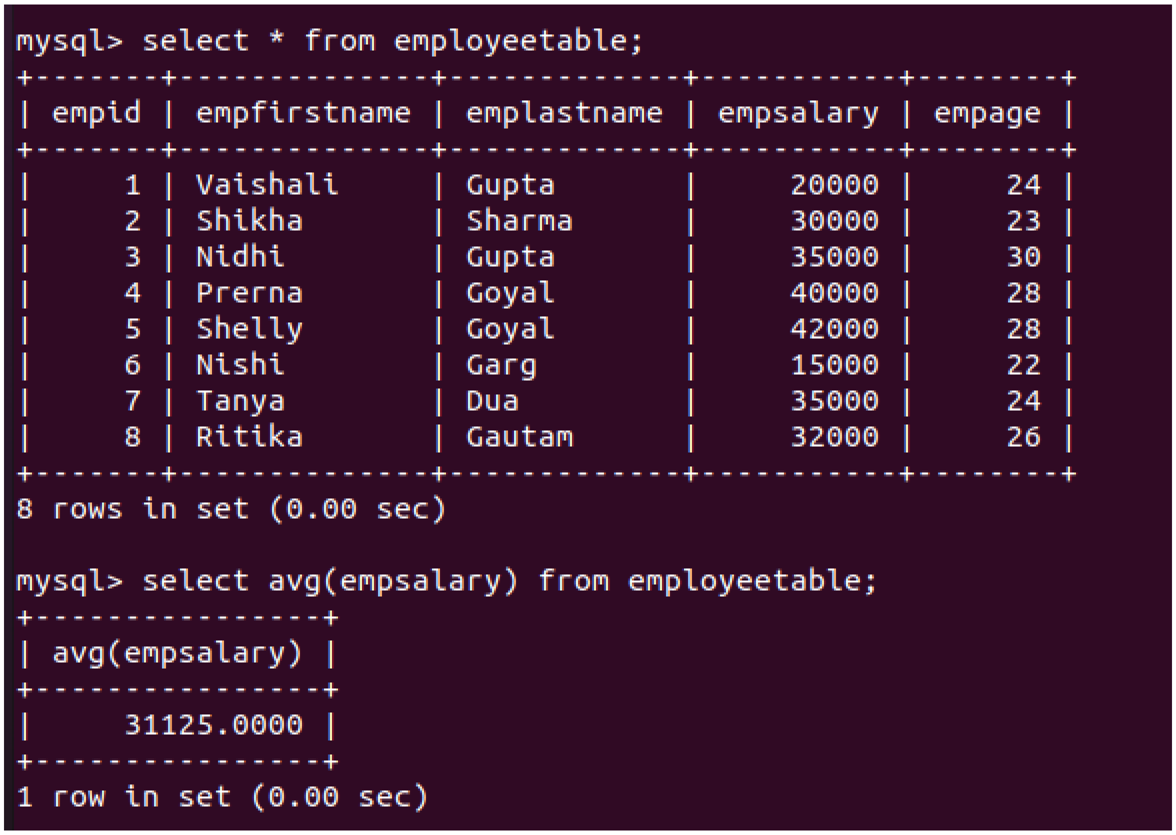
2. **Update salary of all employees by a salary passed as a parameter whose existing salary is less than the average salary.**

<EmployeeRepository.java>

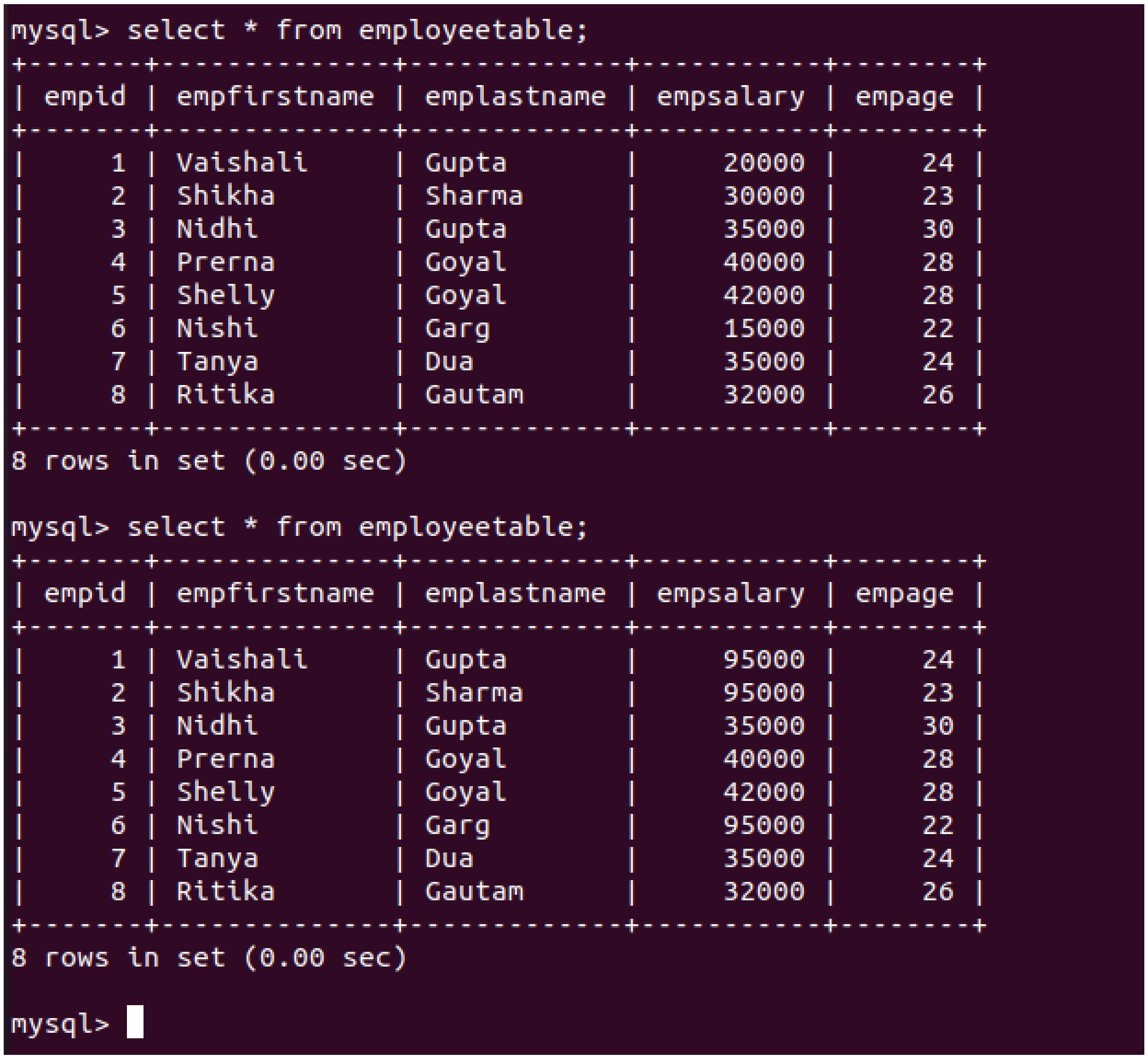
```
@Query("Select avg(salary) from Employee")
int findAvgSalary();


@Query("Update Employee Set salary =:sal where salary <:findAvgSalary")
@Modifying
void updateEmployeesHavingSalaryLessThanAvgSalary(@Param("sal") int sal,
                                   @Param("findAvgSalary") int findAvgSalary);
```

<ApplicationTest.java>

```
@Test
public void testFindAvgSalary(){
  System.out.println(employeeRepository.findAvgSalary());
}


@Test
@Transactional
@Rollback(value = false)
public void testUpdateEmployeesHavingSalaryLessThanAvgSalary(){
  employeeRepository.updateEmployeesHavingSalaryLessThanAvgSalary(85000,
       employeeRepository.findAvgSalary());
}
```



```
mysql> select * from employeetable;
+-------+-------------+------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+-------------+------------+-----------+--------+
|     1 | Vaishali     | Gupta       |     20000 |     24 |
|     2 | Shikha       | Sharma      |     30000 |     23 |
|     3 | Nidhi        | Gupta       |     35000 |     30 |
|     4 | Prerna       | Goyal       |     40000 |     28 |
|     5 | Shelly       | Goyal       |     42000 |     28 |
|     6 | Nishi        | Garg        |     15000 |     22 |
|     7 | Tanya        | Dua         |     35000 |     24 |
|     8 | Ritika       | Gautam      |     32000 |     26 |
+-------+-------------+------------+-----------+--------+
8 rows in set (0.00 sec)

mysql> select avg(empsalary) from employeetable;
+----------------+
| avg(empsalary) |
+----------------+
|     31125.0000 |
+----------------+
1 row in set (0.00 sec)
```



```
mysql> select * from employeetable;
+-------+-------------+------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+-------------+------------+-----------+--------+
|     1 | Vaishali     | Gupta       |     20000 |     24 |
|     2 | Shikha       | Sharma      |     30000 |     23 |
|     3 | Nidhi        | Gupta       |     35000 |     30 |
|     4 | Prerna       | Goyal       |     40000 |     28 |
|     5 | Shelly       | Goyal       |     42000 |     28 |
|     6 | Nishi        | Garg        |     15000 |     22 |
|     7 | Tanya        | Dua         |     35000 |     24 |
|     8 | Ritika       | Gautam      |     32000 |     26 |
+-------+-------------+------------+-----------+--------+
8 rows in set (0.00 sec)

mysql> select * from employeetable;
+-------+-------------+------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+-------------+------------+-----------+--------+
|     1 | Vaishali     | Gupta       |     95000 |     24 |
|     2 | Shikha       | Sharma      |     95000 |     23 |
|     3 | Nidhi        | Gupta       |     35000 |     30 |
|     4 | Prerna       | Goyal       |     40000 |     28 |
|     5 | Shelly       | Goyal       |     42000 |     28 |
|     6 | Nishi        | Garg        |     95000 |     22 |
|     7 | Tanya        | Dua         |     35000 |     24 |
|     8 | Ritika       | Gautam      |     32000 |     26 |
+-------+-------------+------------+-----------+--------+
8 rows in set (0.00 sec)

mysql> 
```

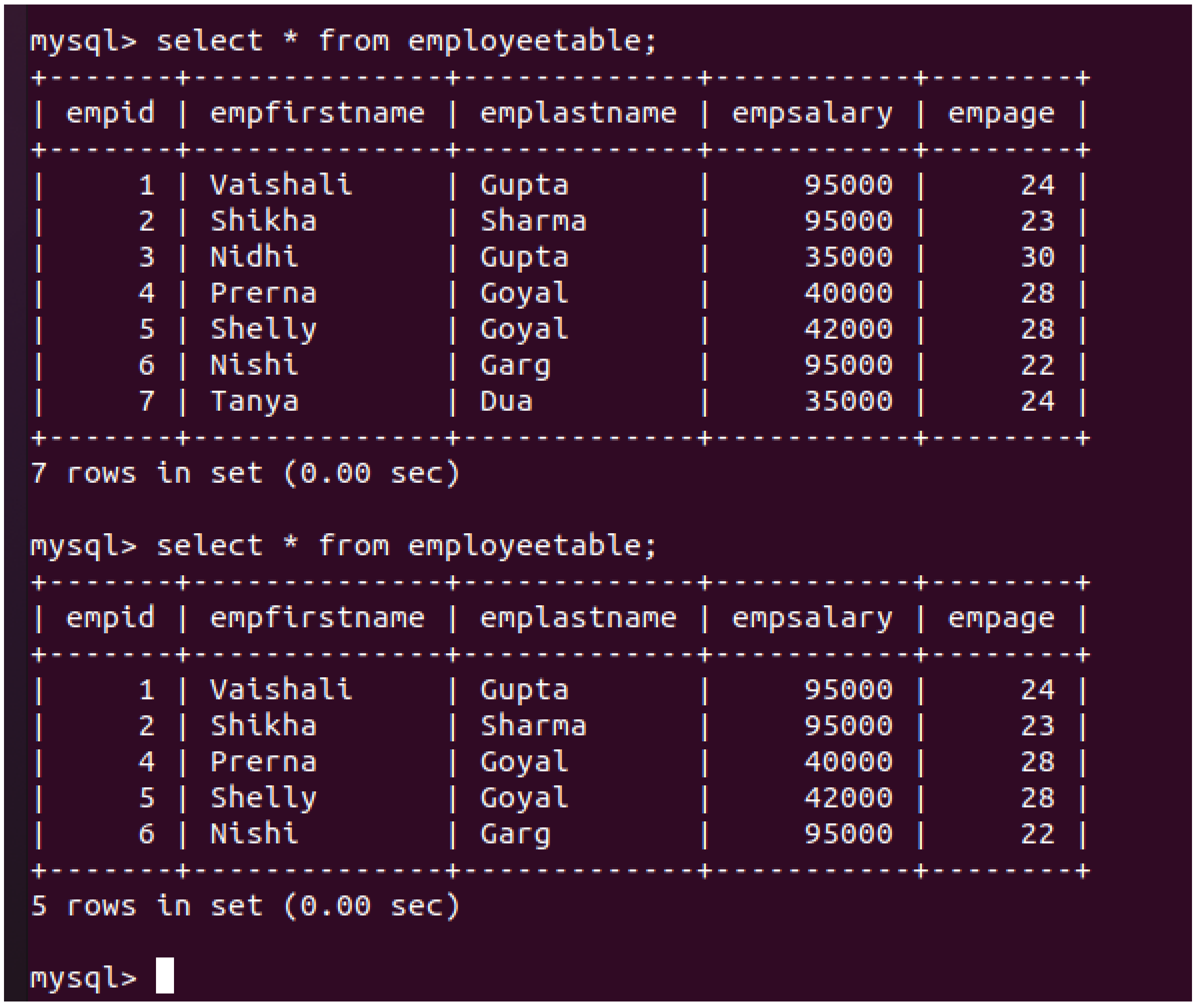
3. **Delete all employees with minimum salary.**

<EmployeeRepository.java>

```
@Query("Select min(salary) from Employee")
int findMinSalary();


@Modifying
@Query("delete from Employee where salary =:findMinSalary")
void deleteEmployeeHavingMinSalary(@Param("findMinSalary") int findMinSalary);
```

<ApplicationTest.java>

```
@Test
public void testFindMinSalary(){
  System.out.println(employeeRepository.findMinSalary());
}

@Test
@Transactional
@Rollback(value = false)
public void testDeleteEmployeesHavingMinSalary(){
  employeeRepository.deleteEmployeeHavingMinSalary(employeeRepository.findMinSalary());
}
```



```
mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Vaishali     | Gupta       |     95000 |     24 |
|     2 | Shikha       | Sharma      |     95000 |     23 |
|     3 | Nidhi        | Gupta       |     35000 |     30 |
|     4 | Prerna       | Goyal       |     40000 |     28 |
|     5 | Shelly       | Goyal       |     42000 |     28 |
|     6 | Nishi        | Garg        |     95000 |     22 |
|     7 | Tanya        | Dua         |     35000 |     24 |
+-------+--------------+-------------+-----------+--------+
7 rows in set (0.00 sec)

mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Vaishali     | Gupta       |     95000 |     24 |
|     2 | Shikha       | Sharma      |     95000 |     23 |
|     4 | Prerna       | Goyal       |     40000 |     28 |
|     5 | Shelly       | Goyal       |     42000 |     28 |
|     6 | Nishi        | Garg        |     95000 |     22 |
+-------+--------------+-------------+-----------+--------+
5 rows in set (0.00 sec)

mysql> 
```

Native SQL Query:

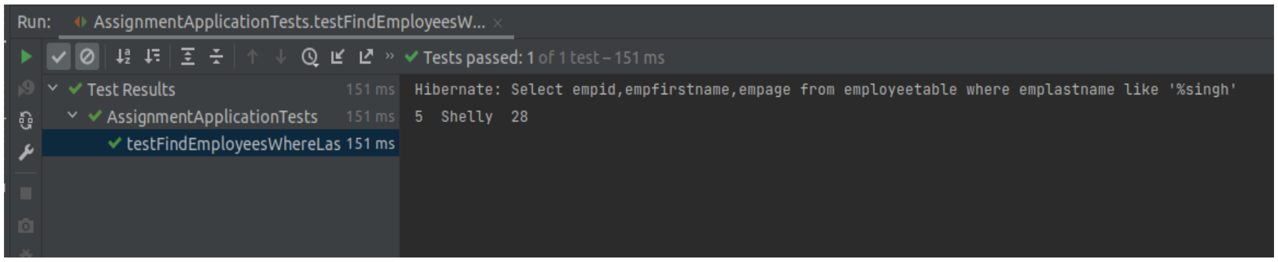
1. **Display the id, first name, age of all employees where last name ends with "singh"**

   <EmployeeRepository.java>

```
@Query(value = "Select empid,empfirstname,empage from employeetable" +

       " where emplastname like '%singh'", nativeQuery = true)
List<Object[]> findEmployeesWhereLastNameEndsWithNQ();
```

   <ApplicationTest.java>

```
@Test
public void testFindEmployeesWhereLastNameEndsWith(){
  List<Object[]> empList =
employeeRepository.findEmployeesWhereLastNameEndsWith();
  for (Object[] emp : empList)
     System.out.println(emp[0] + "  " + emp[1] + "  " +  emp[2]);
}
```



```
Hibernate: Select empid,empfirstname,empage from employeetable where emplastname like '%singh'
5  Shelly  28
```

Run: AssignmentApplicationTests.testFindEmployeesW...
Tests passed: 1 of 1 test – 151 ms
Test Results 151 ms
AssignmentApplicationTests 151 ms
testFindEmployeesWhereLas 151 ms

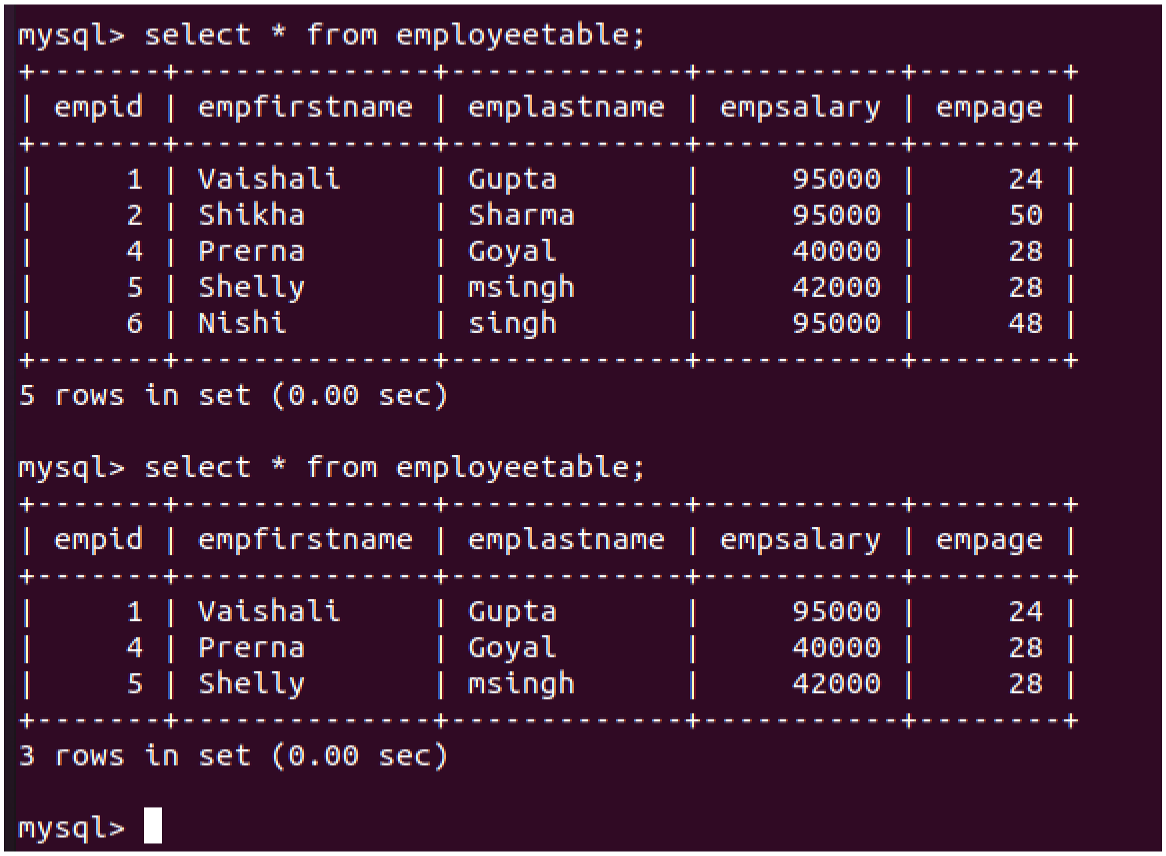
2. **Delete all employees with age greater than 45(Should be passed as a parameter)**

   <EmployeeRepository.java>

```
@Modifying
@Query(value = "Delete from employeetable where empage >:age", nativeQuery = true)
void deleteAllEmployeeWhereAgeGreaterThanNQ(@Param("age") int age);
```

   <ApplicationTest.java>

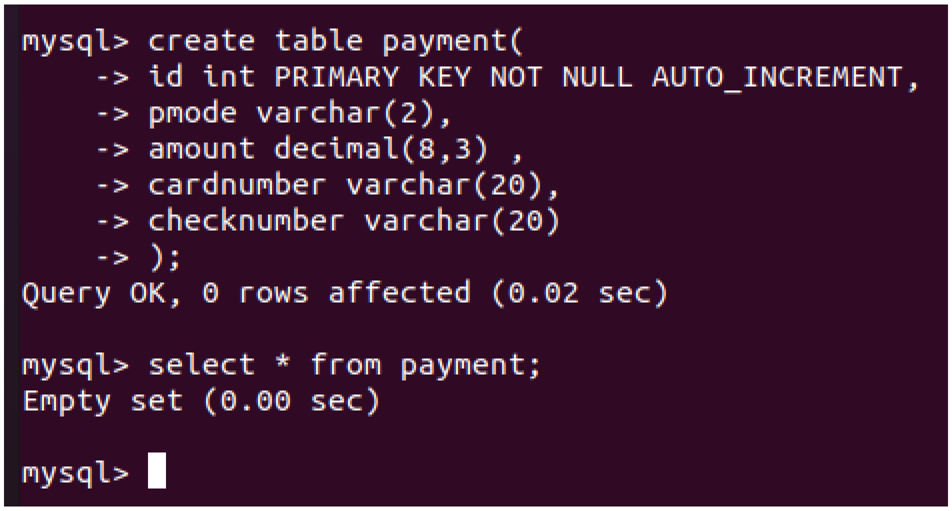
```
@Test
@Transactional
@Rollback(value = false)
public void testDeleteEmployeeWhereAgeGreaterThan(){
  employeeRepository.deleteAllEmployeeWhereAgeGreaterThan(45);
}
```

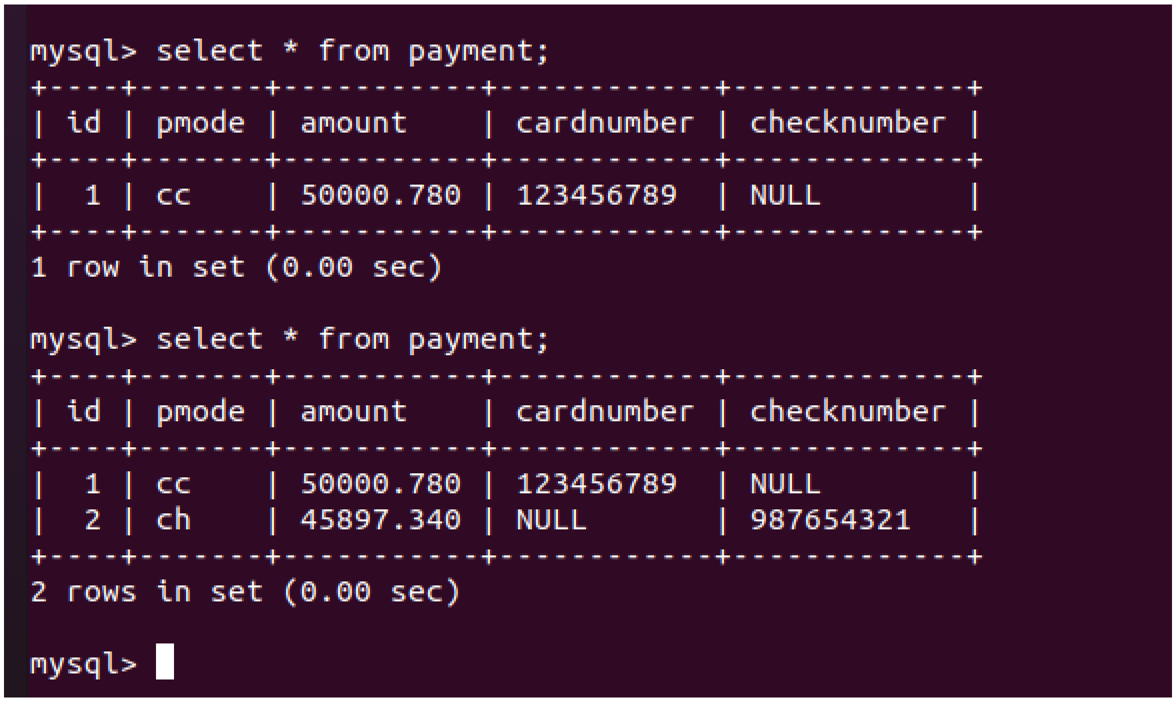


```
mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Vaishali     | Gupta       |     95000 |     24 |
|     2 | Shikha       | Sharma      |     95000 |     50 |
|     4 | Prerna       | Goyal       |     40000 |     28 |
|     5 | Shelly       | msingh      |     42000 |     28 |
|     6 | Nishi        | singh       |     95000 |     48 |
+-------+--------------+-------------+-----------+--------+
5 rows in set (0.00 sec)

mysql> select * from employeetable;
+-------+--------------+-------------+-----------+--------+
| empid | empfirstname | emplastname | empsalary | empage |
+-------+--------------+-------------+-----------+--------+
|     1 | Vaishali     | Gupta       |     95000 |     24 |
|     4 | Prerna       | Goyal       |     40000 |     28 |
|     5 | Shelly       | msingh      |     42000 |     28 |
+-------+--------------+-------------+-----------+--------+
3 rows in set (0.00 sec)

mysql> 
```

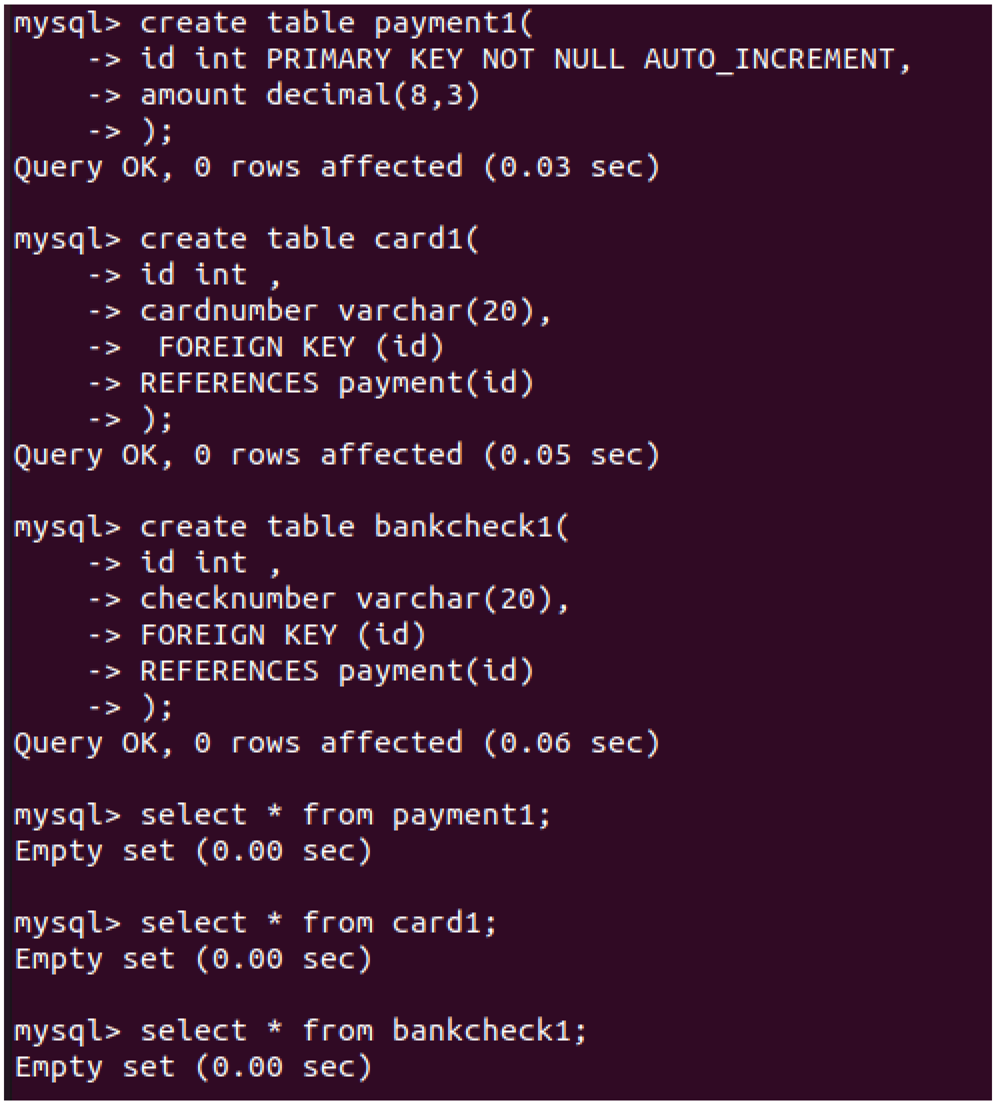
Inheritance Mapping:

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

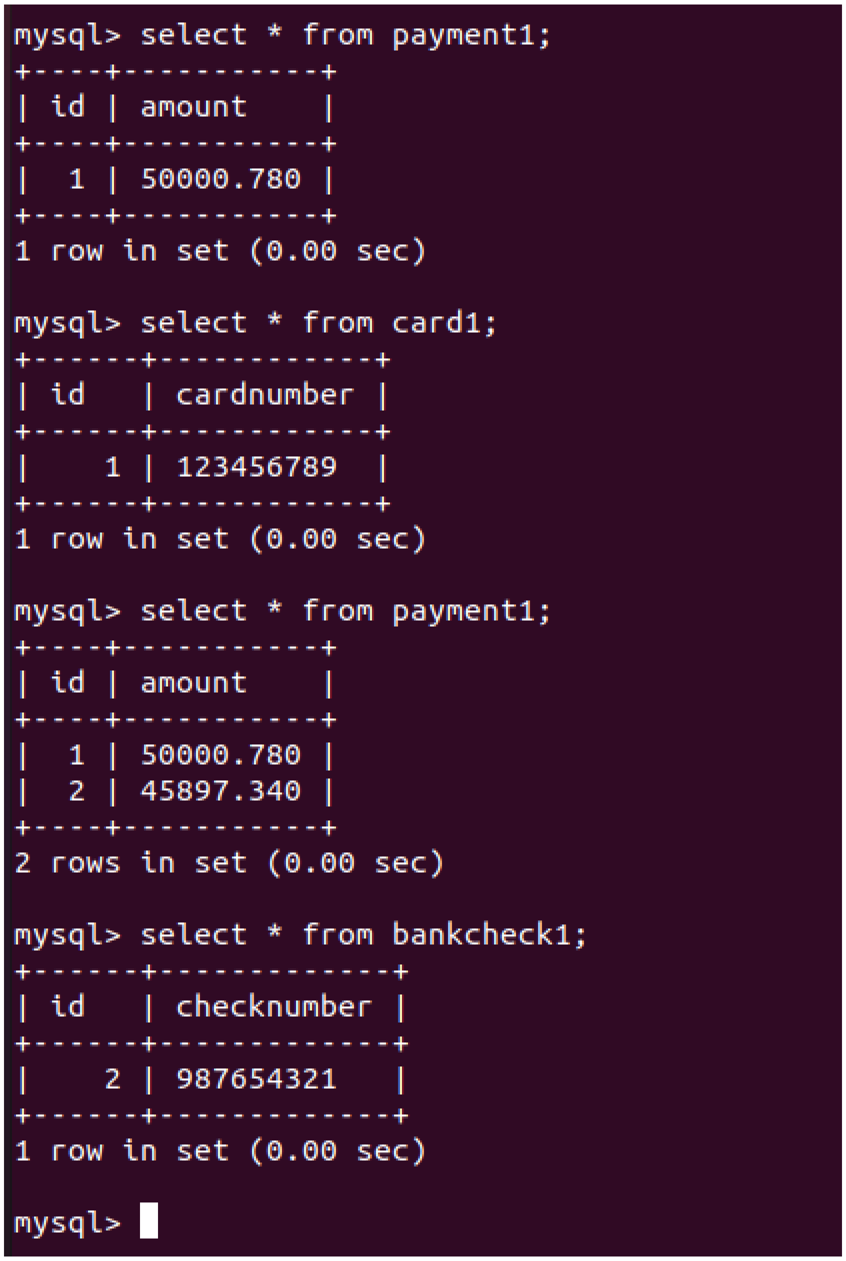

```
mysql> create table payment(
    -> id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> pmode varchar(2),
    -> amount decimal(8,3) ,
    -> cardnumber varchar(20),
    -> checknumber varchar(20)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> select * from payment;
Empty set (0.00 sec)

mysql> 
```


```
mysql> select * from payment;
+----+-------+-----------+------------+-------------+
| id | pmode | amount    | cardnumber | checknumber |
+----+-------+-----------+------------+-------------+
|  1 | cc    | 50000.780 | 123456789  | NULL        |
+----+-------+-----------+------------+-------------+
1 row in set (0.00 sec)

mysql> select * from payment;
+----+-------+-----------+------------+-------------+
| id | pmode | amount    | cardnumber | checknumber |
+----+-------+-----------+------------+-------------+
|  1 | cc    | 50000.780 | 123456789  | NULL        |
|  2 | ch    | 45897.340 | NULL       | 987654321   |
+----+-------+-----------+------------+-------------+
2 rows in set (0.00 sec)

mysql> 
```

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

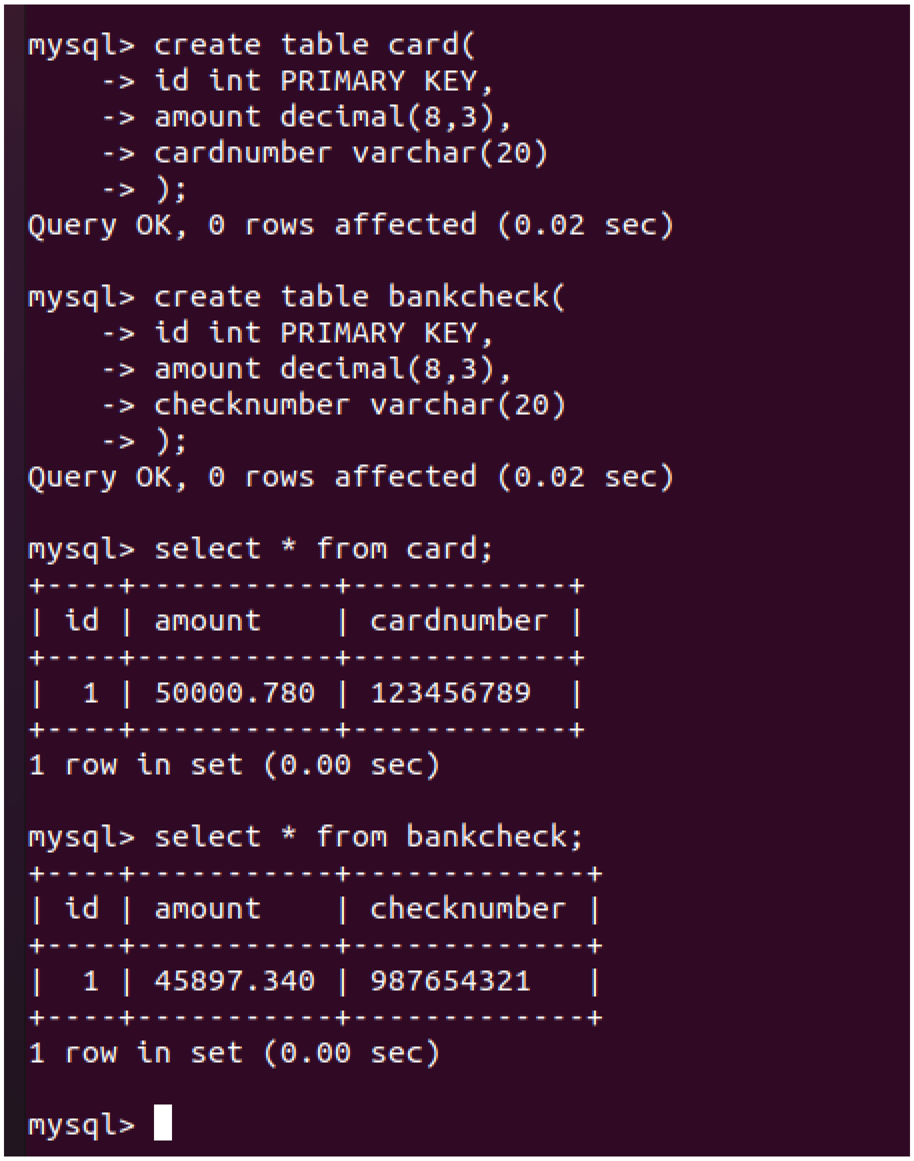


```
mysql> create table payment1(
    -> id int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    -> amount decimal(8,3)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> create table card1(
    -> id int ,
    -> cardnumber varchar(20),
    ->  FOREIGN KEY (id)
    -> REFERENCES payment(id)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> create table bankcheck1(
    -> id int ,
    -> checknumber varchar(20),
    -> FOREIGN KEY (id)
    -> REFERENCES payment(id)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> select * from payment1;
Empty set (0.00 sec)

mysql> select * from card1;
Empty set (0.00 sec)

mysql> select * from bankcheck1;
Empty set (0.00 sec)
```



```
mysql> select * from payment1;
+----+-----------+
| id | amount    |
+----+-----------+
|  1 | 50000.780 |
+----+-----------+
1 row in set (0.00 sec)

mysql> select * from card1;
+------+------------+
| id   | cardnumber |
+------+------------+
|    1 | 123456789  |
+------+------------+
1 row in set (0.00 sec)

mysql> select * from payment1;
+----+-----------+
| id | amount    |
+----+-----------+
|  1 | 50000.780 |
|  2 | 45897.340 |
+----+-----------+
2 rows in set (0.00 sec)

mysql> select * from bankcheck1;
+------+-------------+
| id   | checknumber |
+------+-------------+
|    2 | 987654321   |
+------+-------------+
1 row in set (0.00 sec)

mysql>
```

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



```
mysql> create table card(
    -> id int PRIMARY KEY,
    -> amount decimal(8,3),
    -> cardnumber varchar(20)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> create table bankcheck(
    -> id int PRIMARY KEY,
    -> amount decimal(8,3),
    -> checknumber varchar(20)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> select * from card;
+----+-----------+------------+
| id | amount    | cardnumber |
+----+-----------+------------+
|  1 | 50000.780 | 123456789  |
+----+-----------+------------+
1 row in set (0.00 sec)

mysql> select * from bankcheck;
+----+-----------+-------------+
| id | amount    | checknumber |
+----+-----------+-------------+
|  1 | 45897.340 | 987654321   |
+----+-----------+-------------+
1 row in set (0.00 sec)

mysql> 
```

<Payment.java>

```
package com.springdatajpawithhibernatepart2.Assignment.inheritance.entities;

import javax.persistence.*;

@Entity
/******** Single table strategy *****************/
//@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
//@DiscriminatorColumn(name = "pmode", discriminatorType = DiscriminatorType.STRING)

/******** Table Per Class Strategy *****************/
```

```
//@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)

/******** Joined strategy *****************/
@Inheritance(strategy = InheritanceType.JOINED)
@Table(name = "payment1")
public class Payment {

  @Id
 //  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  private double amount;

  public Long getId() {
      return id;
  }

  public void setId(Long id) {
      this.id = id;
  }

  public double getAmount() {
      return amount;
  }

  public void setAmount(double amount) {
      this.amount = amount;
  }

}
```

<CreditCard.java>

```
package com.springdatajpawithhibernatepart2.Assignment.inheritance.entities;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

@Entity
/******** Single table strategy *****************/
```

```
//@DiscriminatorValue("cc")

/******** Table Per Class Strategy ******************/
//@Table(name = "card")

/******** Joined strategy ******************/
@Table(name = "card1")
@PrimaryKeyJoinColumn(name = "id")
public class CreditCard extends Payment{
  private String cardnumber;

  public String getCardnumber() {
     return cardnumber;
  }

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

<Check.java>

```
package com.springdatajpawithhibernatepart2.Assignment.inheritance.entities;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

@Entity
/******** Single table strategy ******************/
//@DiscriminatorValue("ch")

/******** Table Per Class Strategy ******************/
//@Table(name = "bankcheck")

/******** Joined strategy ******************/
@Table(name = "bankcheck1")
@PrimaryKeyJoinColumn(name = "id")
public class Check extends Payment{
  private String checknumber;

  public String getChecknumber() {
      return checknumber;
  }

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

Component Mapping:

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

<EmployeeMapping.java>

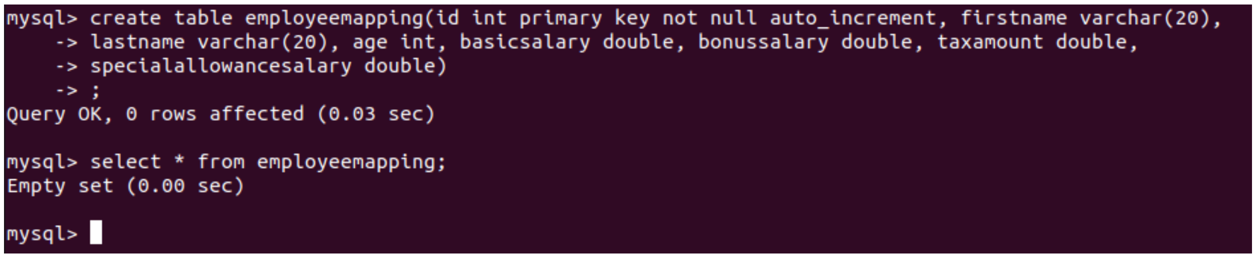
```
package com.springdatajpawithhibernatepart2.Assignment.componentmapping.entities;

import javax.persistence.*;

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

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    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) {
        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;
    }

    @Override
    public String toString() {
        return "EmployeeMapping{" +
                "id=" + id +
                ", firstname='" + firstname + '\'' +
                ", lastname='" + lastname + '\'' +
                ", age=" + age +
                ", salary=" + salary +
                '}';
    }
}
```

<Salary.java>

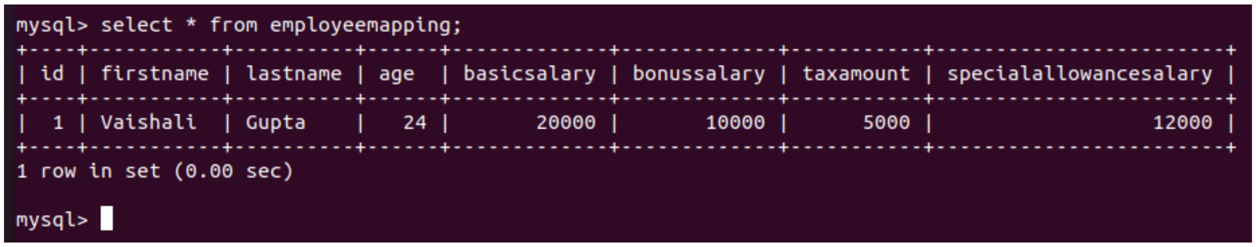
```
package com.springdatajpawithhibernatepart2.Assignment.componentmapping.entities;

import javax.persistence.Column;
import javax.persistence.Embeddable;

@Embeddable
public class Salary {
  @Column(name = "basicsalary")
  private double basicSalary;

  @Column(name = "bonussalary")
  private double bonusSalary;

  @Column(name = "taxamount")
  private double taxAmount;

  @Column(name = "specialallowancesalary")
  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;
  }
}
```

<ApplicationTest.java>

```
@Test
public void testCreateEmployeeMapping(){
  EmployeeMapping employee = new EmployeeMapping();
  employee.setAge(24);
  employee.setFirstname("Vaishali");
  employee.setLastname("Gupta");

  Salary salary = new Salary();
  salary.setBasicSalary(20000);
  salary.setBonusSalary(10000);
  salary.setSpecialAllowanceSalary(12000);
  salary.setTaxAmount(5000);

  employee.setSalary(salary);

  employeeMappingRepository.save(employee);

}
```



```
mysql> create table employeemapping(id int primary key not null auto_increment, firstname varchar(20),
    -> lastname varchar(20), age int, basicsalary double, bonussalary double, taxamount double,
    -> specialallowancesalary double)
    -> ;
Query OK, 0 rows affected (0.03 sec)

mysql> select * from employeemapping;
Empty set (0.00 sec)

mysql> 
```



```
mysql> select * from employeemapping;
+----+-----------+----------+------+-------------+-------------+-----------+------------------------+
| id | firstname | lastname | age  | basicsalary | bonussalary | taxamount | specialallowancesalary |
+----+-----------+----------+------+-------------+-------------+-----------+------------------------+
|  1 | Vaishali  | Gupta    |   24 |       20000 |       10000 |      5000 |                  12000 |
+----+-----------+----------+------+-------------+-------------+-----------+------------------------+
1 row in set (0.00 sec)

mysql> 
```