Assignment - Spring Data JPA - Part 2

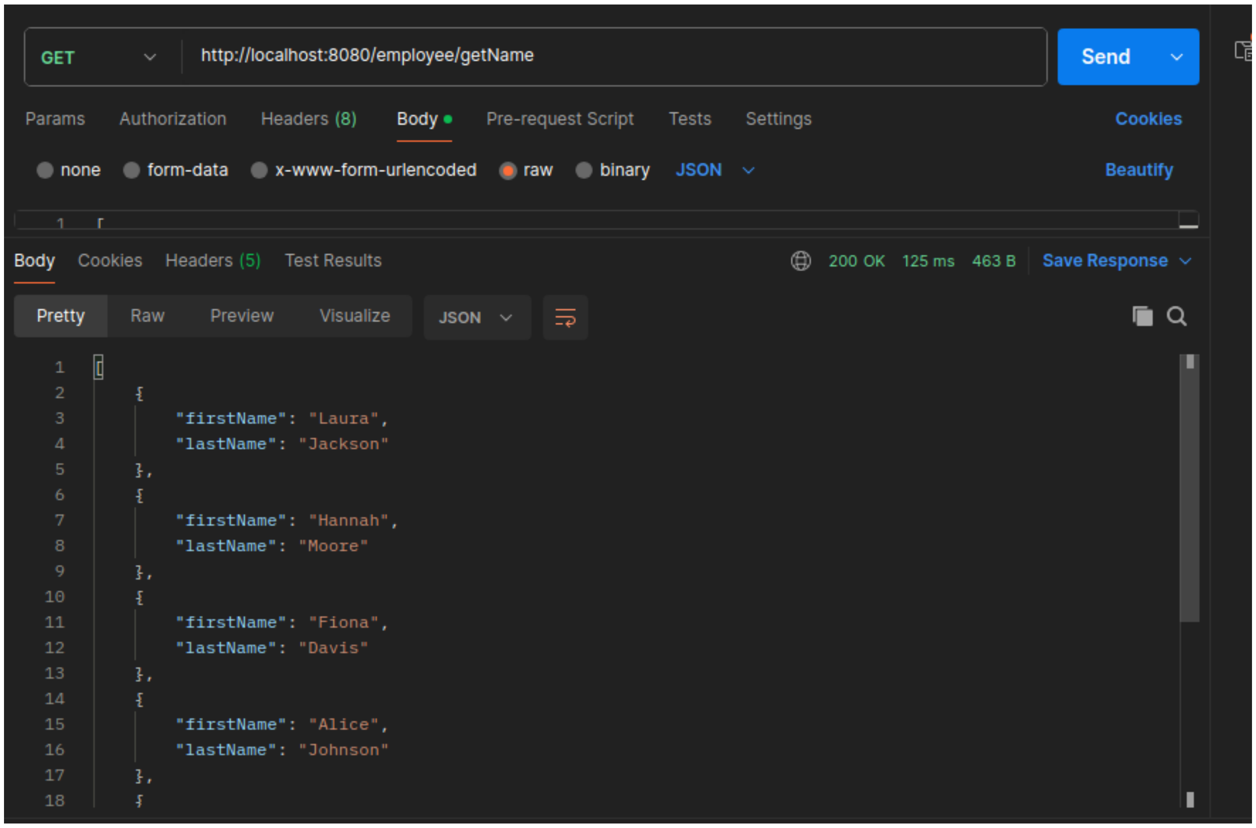
JPQL: Instructions:

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

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

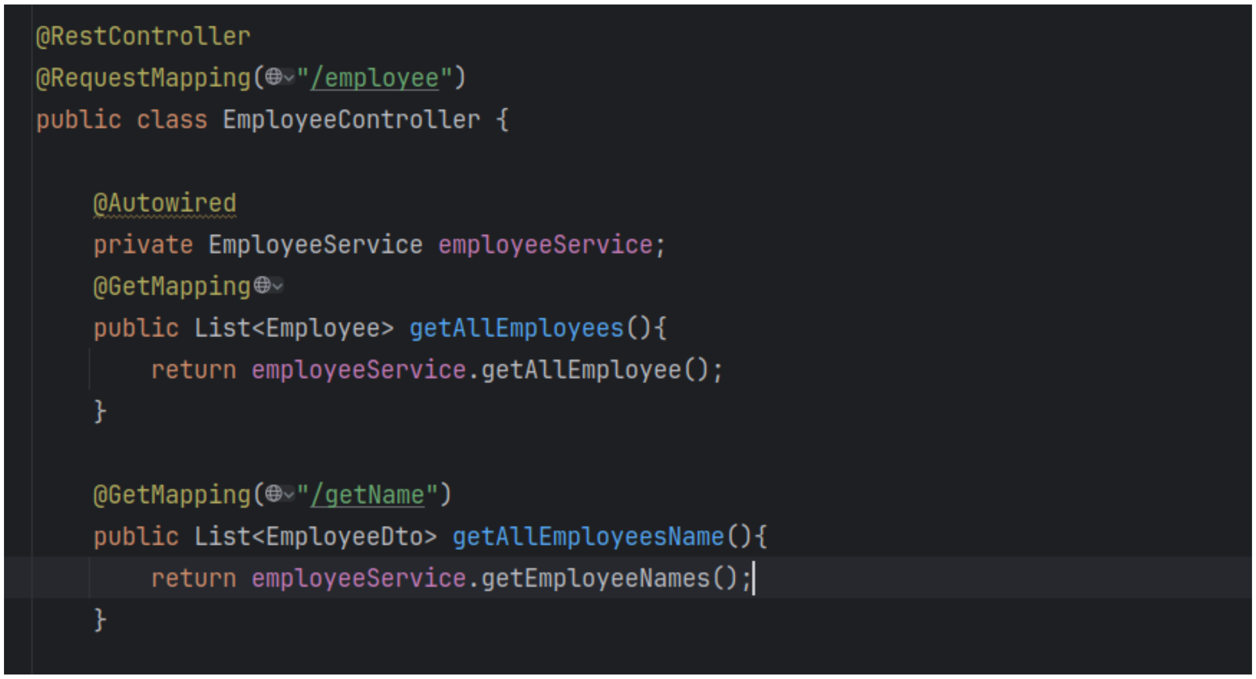
Questions:

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.

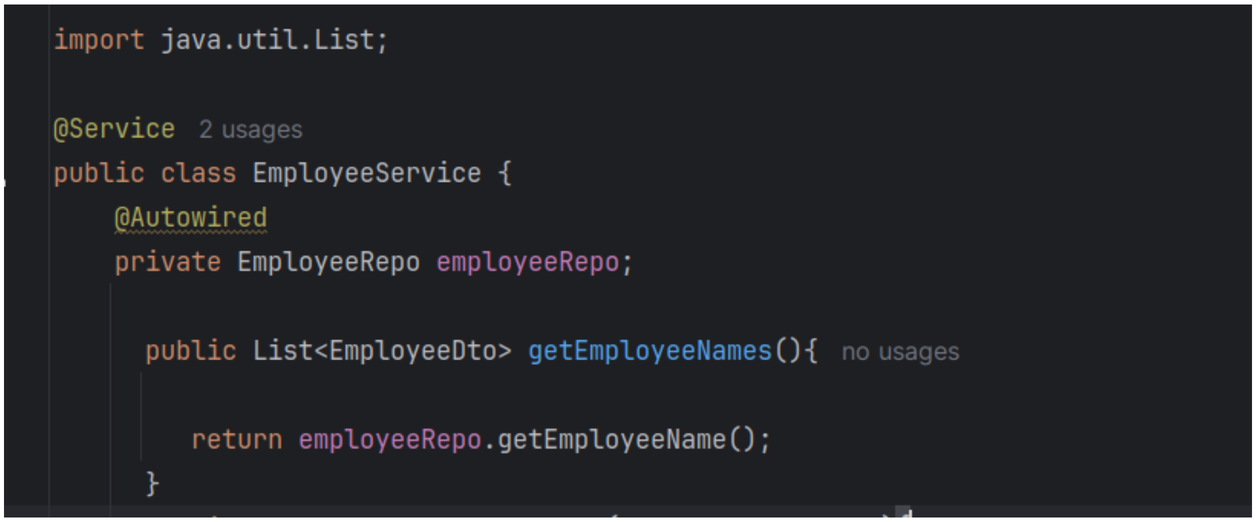


http://localhost:8080/employee/getName

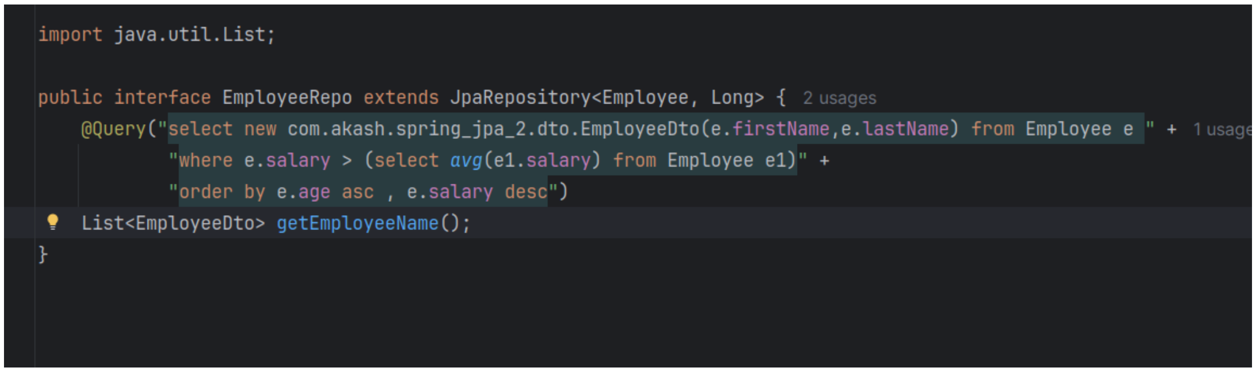
```
[
    {
        "firstName": "Laura",
        "lastName": "Jackson"
    },
    {
        "firstName": "Hannah",
        "lastName": "Moore"
    },
    {
        "firstName": "Fiona",
        "lastName": "Davis"
    },
    {
        "firstName": "Alice",
        "lastName": "Johnson"
    },
    {
```



```java
@RestController
@RequestMapping("/employee")
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;
    @GetMapping
    public List<Employee> getAllEmployees(){
        return employeeService.getAllEmployee();
    }

    @GetMapping("/getName")
    public List<EmployeeDto> getAllEmployeesName(){
        return employeeService.getEmployeeNames();
    }
```

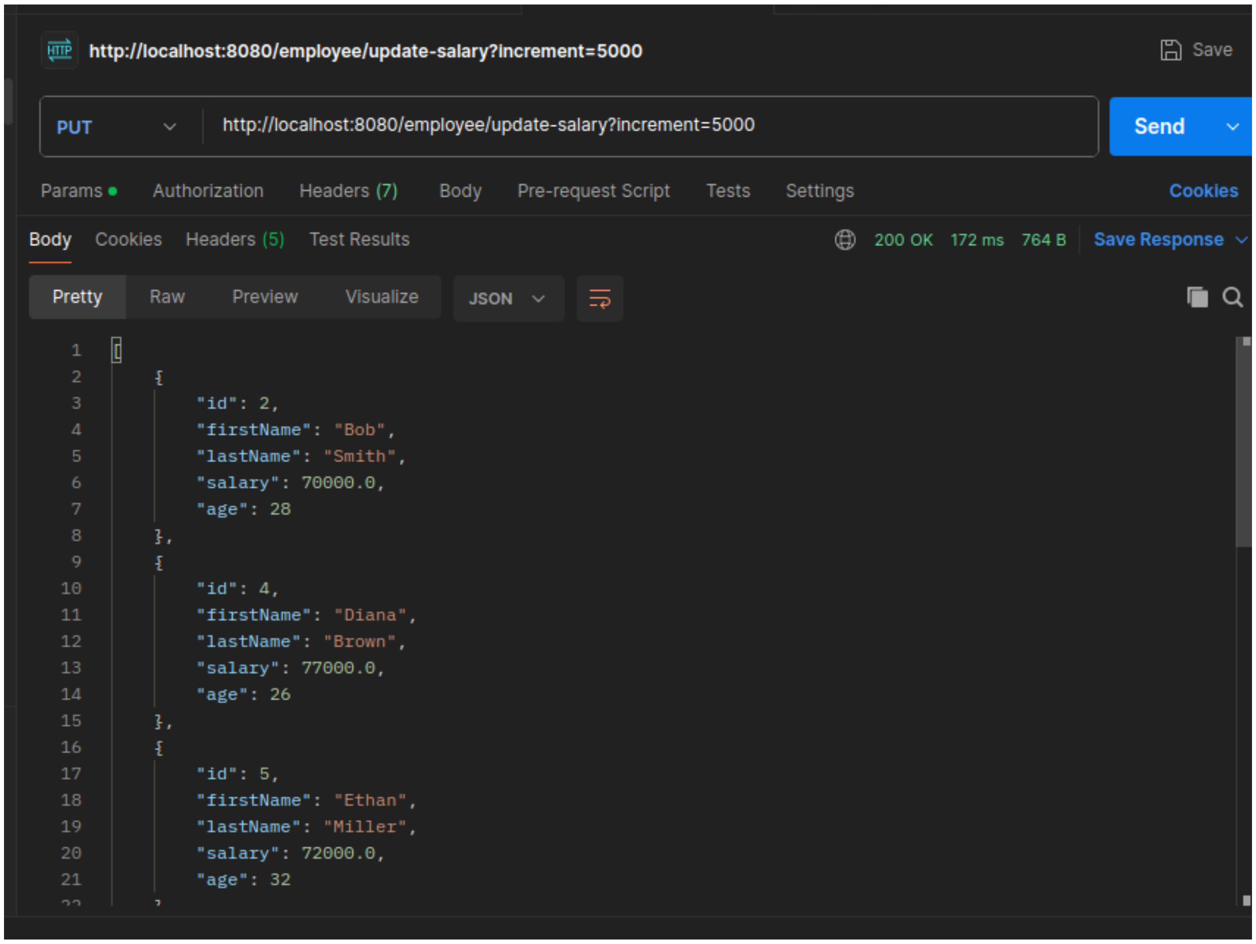


```java
import java.util.List;

@Service  2 usages
public class EmployeeService {
    @Autowired
    private EmployeeRepo employeeRepo;

    public List<EmployeeDto> getEmployeeNames(){  no usages

        return employeeRepo.getEmployeeName();
    }
```



```java
import java.util.List;

public interface EmployeeRepo extends JpaRepository<Employee, Long> {  2 usages
    @Query("select new com.akash.spring_jpa_2.dto.EmployeeDto(e.firstName,e.lastName) from Employee e " +  1 usage
            "where e.salary > (select avg(e1.salary) from Employee e1)" +
            "order by e.age asc , e.salary desc")
    List<EmployeeDto> getEmployeeName();
}
```

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

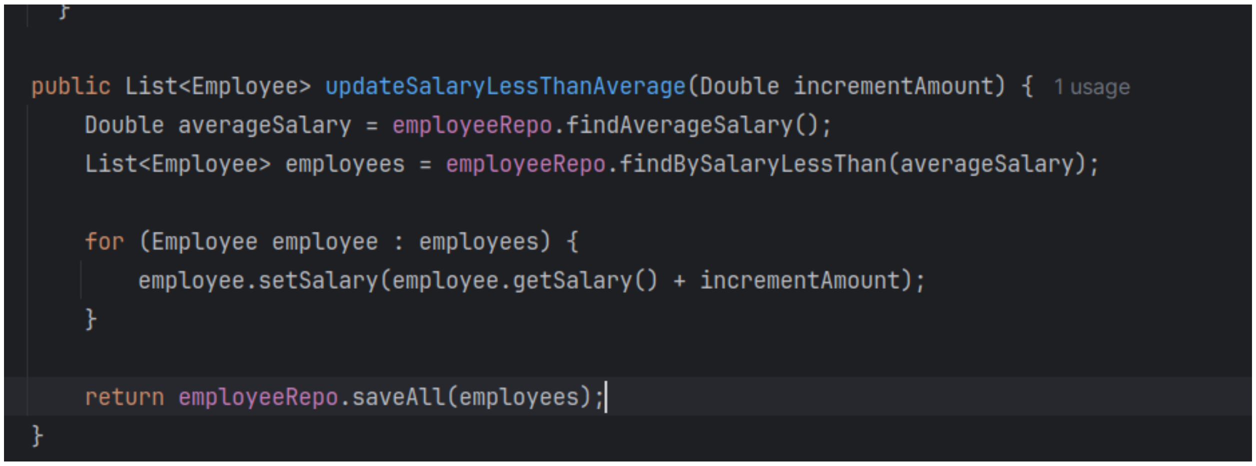


http://localhost:8080/employee/update-salary?increment=5000

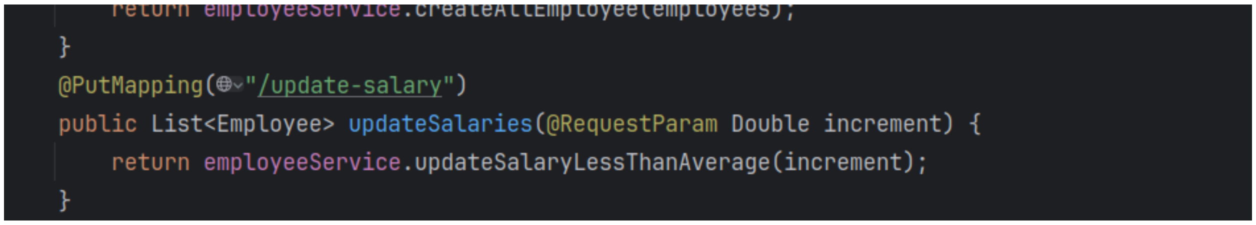
PUT http://localhost:8080/employee/update-salary?increment=5000

200 OK 172 ms 764 B

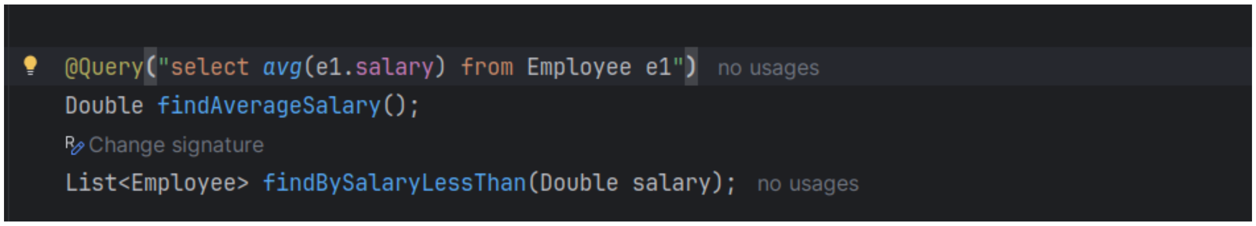
```json
[
    {
        "id": 2,
        "firstName": "Bob",
        "lastName": "Smith",
        "salary": 70000.0,
        "age": 28
    },
    {
        "id": 4,
        "firstName": "Diana",
        "lastName": "Brown",
        "salary": 77000.0,
        "age": 26
    },
    {
        "id": 5,
        "firstName": "Ethan",
        "lastName": "Miller",
        "salary": 72000.0,
        "age": 32
```



```java
}

public List<Employee> updateSalaryLessThanAverage(Double incrementAmount) {  1 usage
    Double averageSalary = employeeRepo.findAverageSalary();
    List<Employee> employees = employeeRepo.findBySalaryLessThan(averageSalary);

    for (Employee employee : employees) {
        employee.setSalary(employee.getSalary() + incrementAmount);
    }

    return employeeRepo.saveAll(employees);
}
```

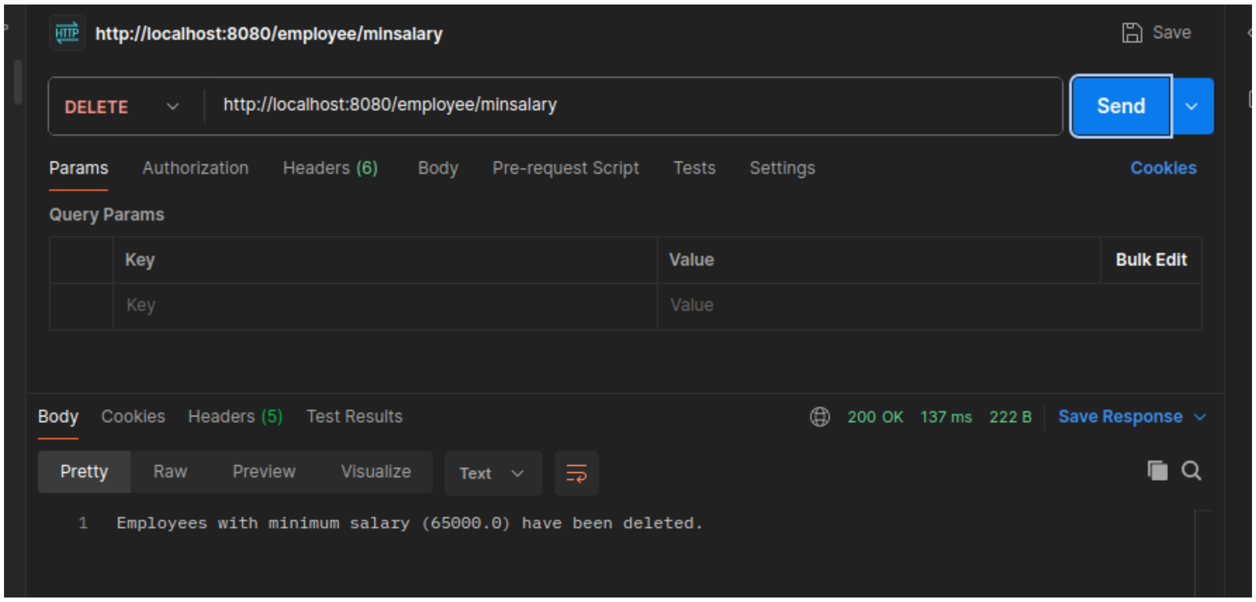


```
    return employeeService.createAllEmployee(employees);
}
@PutMapping("/update-salary")
public List<Employee> updateSalaries(@RequestParam Double increment) {
    return employeeService.updateSalaryLessThanAverage(increment);
}
```

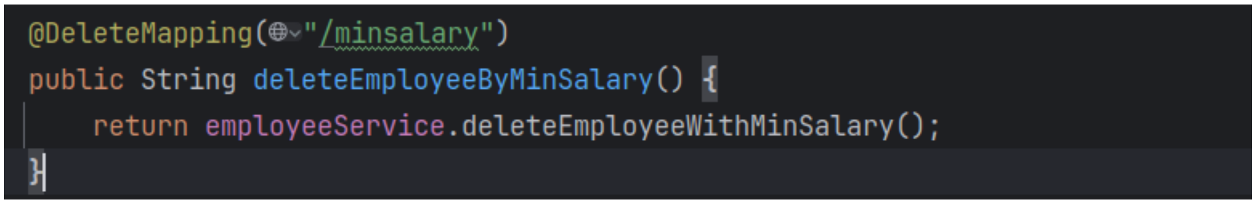


```
@Query("select avg(e1.salary) from Employee e1")  no usages
Double findAverageSalary();
Change signature
List<Employee> findBySalaryLessThan(Double salary);  no usages
```

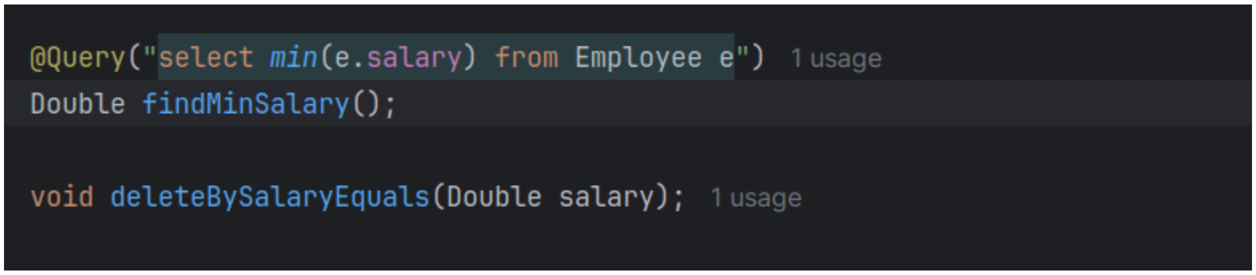
3) Delete all employees with minimum salary.



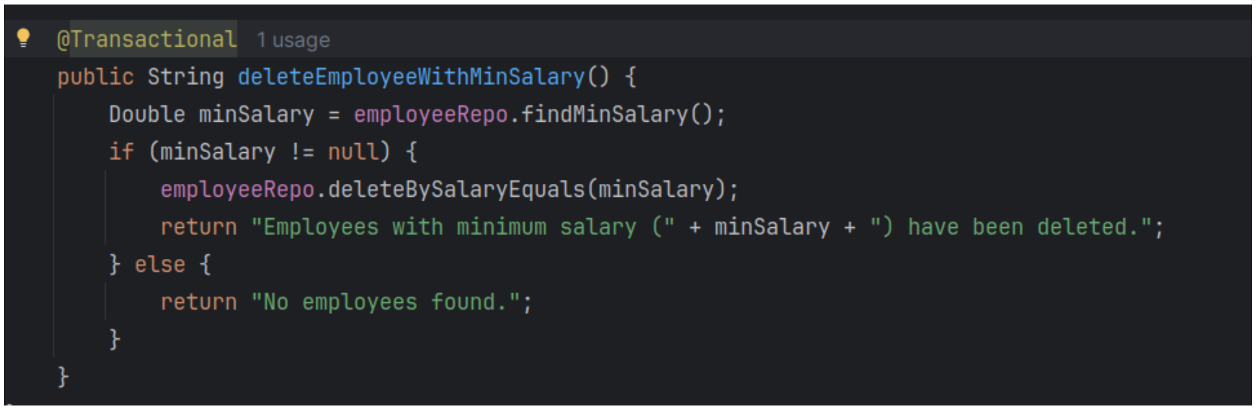
```
Employees with minimum salary (65000.0) have been deleted.
```



```
@DeleteMapping("/minsalary")
public String deleteEmployeeByMinSalary() {
    return employeeService.deleteEmployeeWithMinSalary();
}
```



```
@Query("select min(e.salary) from Employee e")  1 usage
Double findMinSalary();

void deleteBySalaryEquals(Double salary);  1 usage
```



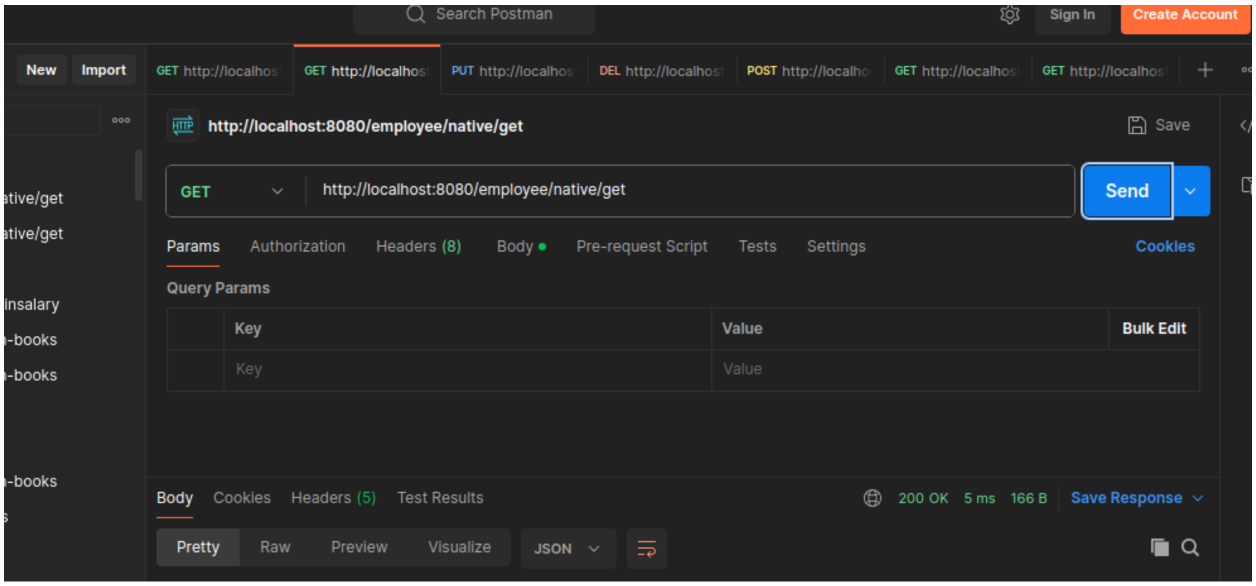
```
@Transactional  1 usage
public String deleteEmployeeWithMinSalary() {
    Double minSalary = employeeRepo.findMinSalary();
    if (minSalary != null) {
        employeeRepo.deleteBySalaryEquals(minSalary);
        return "Employees with minimum salary (" + minSalary + ") have been deleted.";
    } else {
        return "No employees found.";
    }
}
```

Q2)Native Query: Instructions:

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

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

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



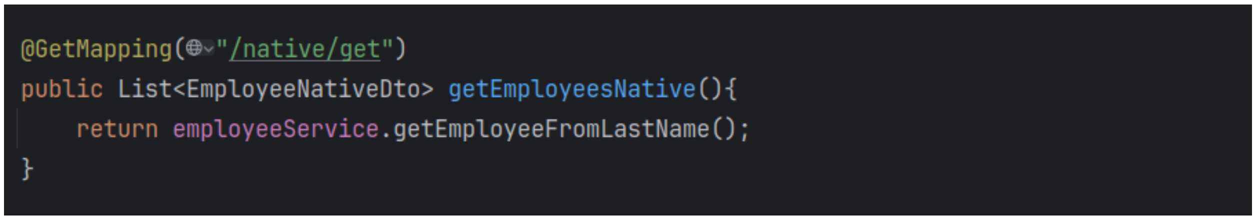
```
GET http://localhost:8080/employee/native/get
```

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

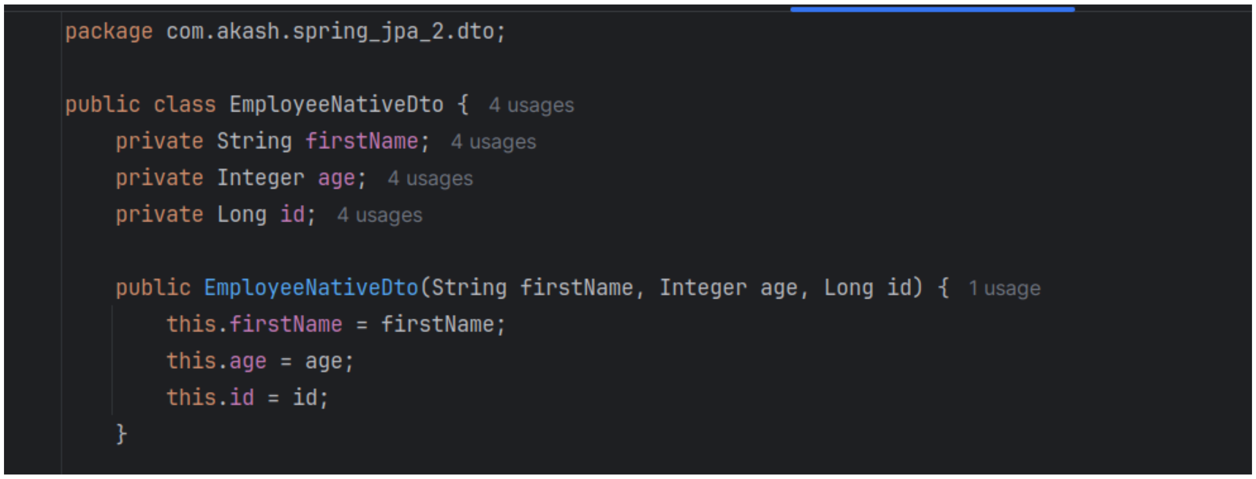
Query Params

Key	Value	Bulk Edit
Key	Value	

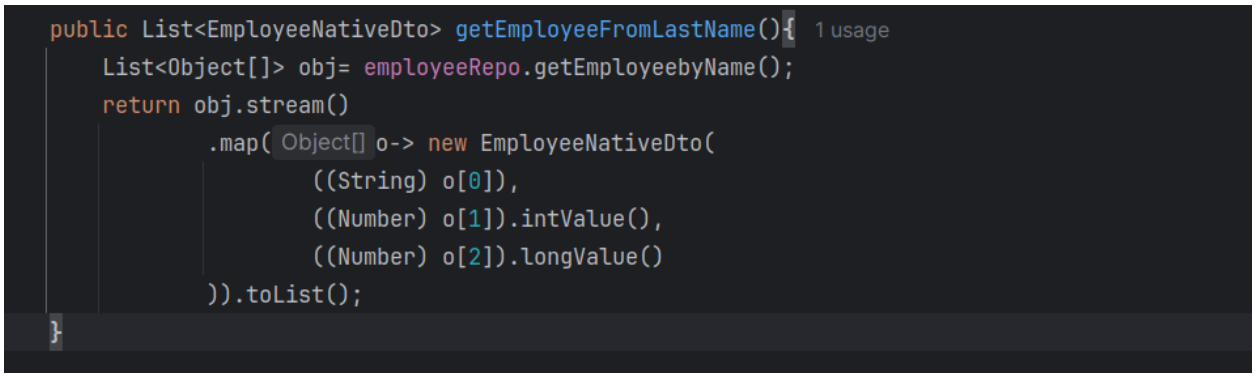
Body | Cookies | Headers (5) | Test Results — 200 OK 5 ms 166 B — Save Response



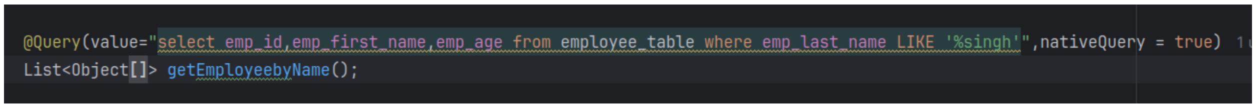
```
@GetMapping("/native/get")
public List<EmployeeNativeDto> getEmployeesNative(){
    return employeeService.getEmployeeFromLastName();
}
```



```
package com.akash.spring_jpa_2.dto;

public class EmployeeNativeDto {  4 usages
    private String firstName;  4 usages
    private Integer age;  4 usages
    private Long id;  4 usages

    public EmployeeNativeDto(String firstName, Integer age, Long id) {  1 usage
        this.firstName = firstName;
        this.age = age;
        this.id = id;
    }
```

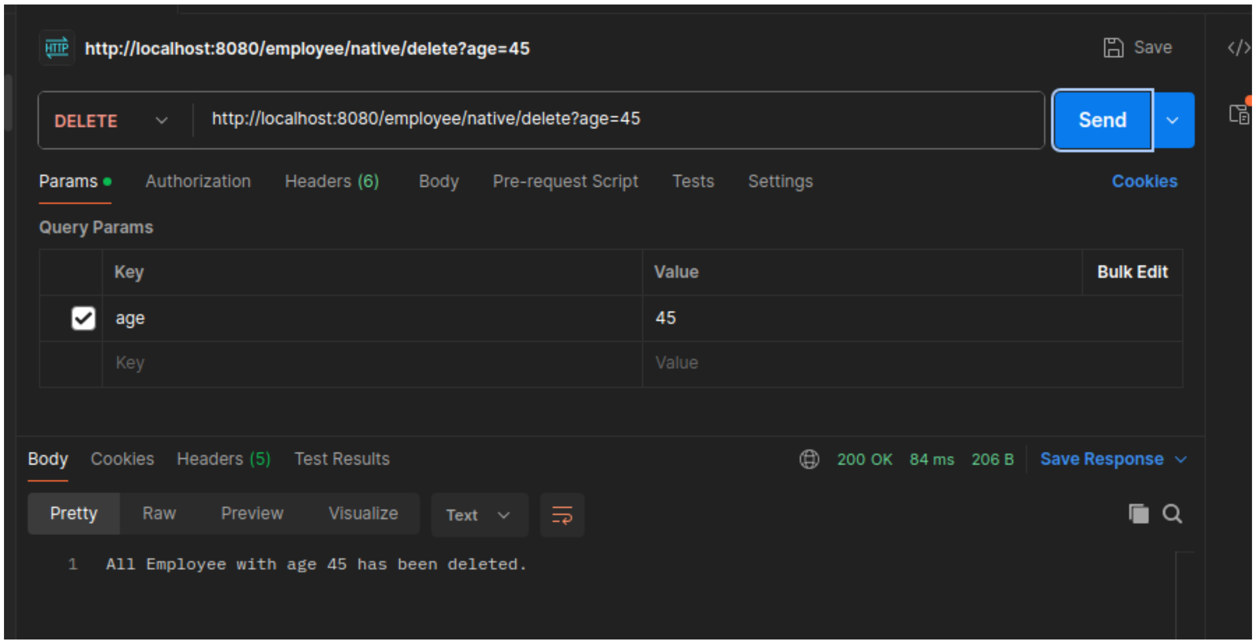


```
public List<EmployeeNativeDto> getEmployeeFromLastName(){  1 usage
    List<Object[]> obj= employeeRepo.getEmployeebyName();
    return obj.stream()
            .map( Object[] o-> new EmployeeNativeDto(
                    ((String) o[0]),
                    ((Number) o[1]).intValue(),
                    ((Number) o[2]).longValue()
            )).toList();
}
```



```
@Query(value="select emp_id,emp_first_name,emp_age from employee_table where emp_last_name LIKE '%singh'",nativeQuery = true)
List<Object[]> getEmployeebyName();
```

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


http://localhost:8080/employee/native/delete?age=45

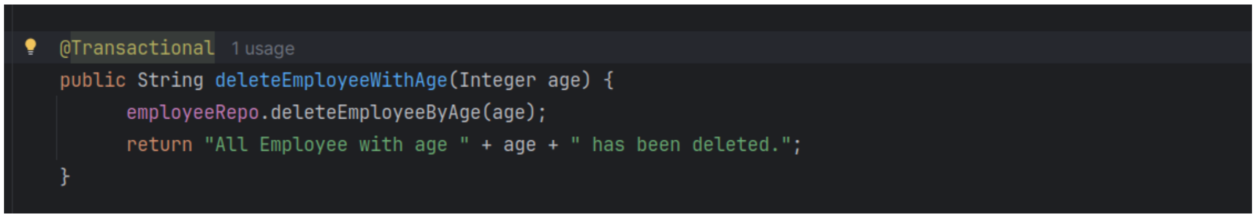
DELETE http://localhost:8080/employee/native/delete?age=45

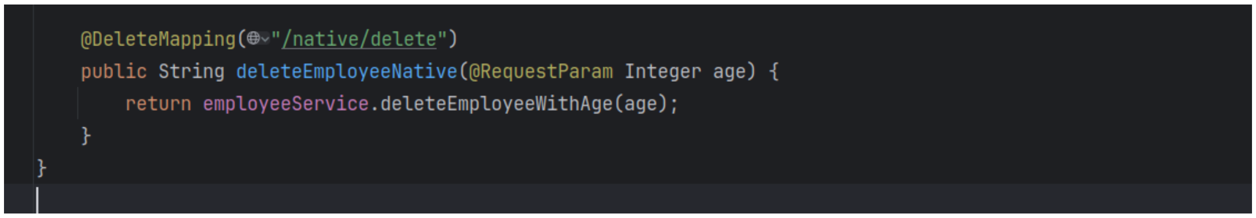
Query Params

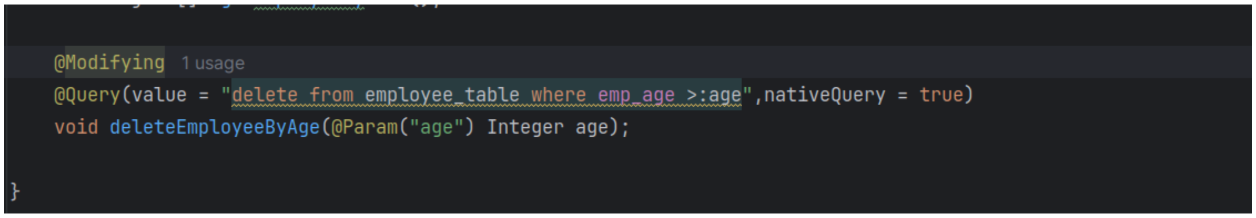
Key	Value
✓ age	45

200 OK 84 ms 206 B

```
All Employee with age 45 has been deleted.
```

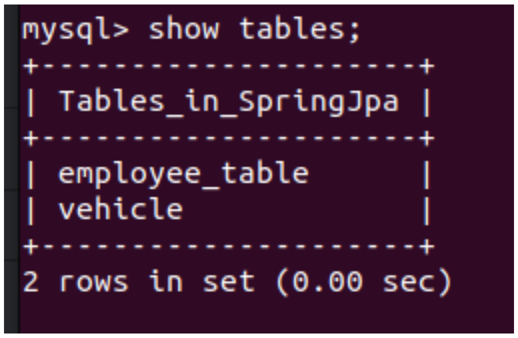

```
@Transactional
public String deleteEmployeeWithAge(Integer age) {
    employeeRepo.deleteEmployeeByAge(age);
    return "All Employee with age " + age + " has been deleted.";
}
```


```
    @DeleteMapping("/native/delete")
    public String deleteEmployeeNative(@RequestParam Integer age) {
        return employeeService.deleteEmployeeWithAge(age);
    }
}
```

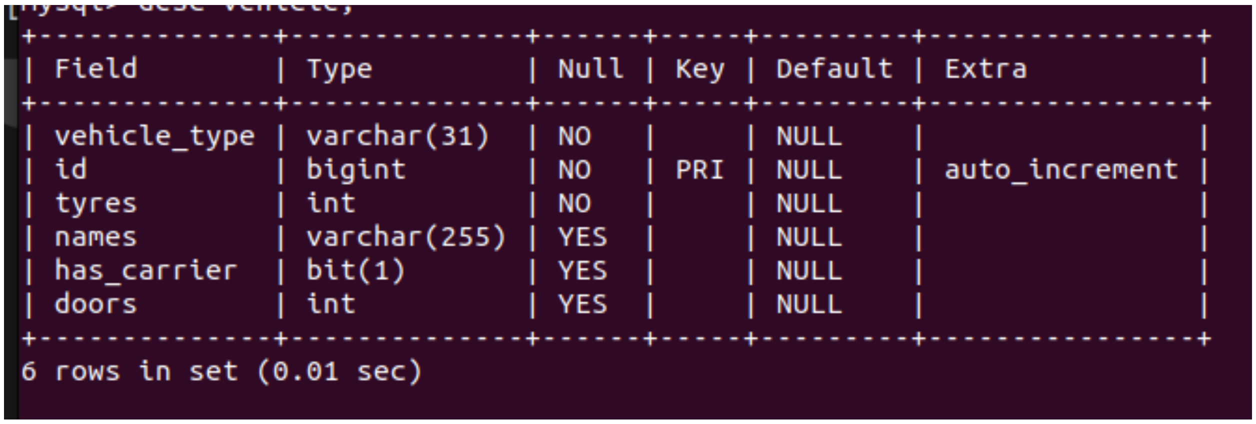

```
    @Modifying
    @Query(value = "delete from employee_table where emp_age >:age",nativeQuery = true)
    void deleteEmployeeByAge(@Param("age") Integer age);

}
```

Q3)Inheritance Mapping:

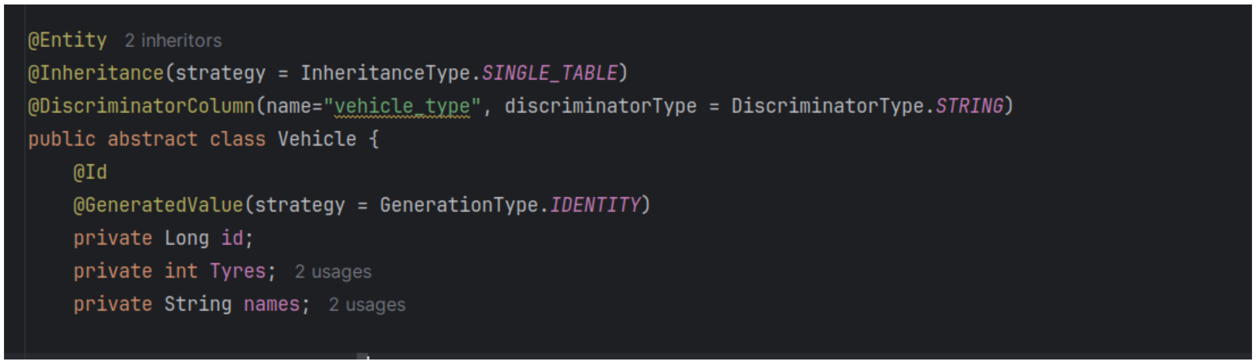
1) Implement and demonstrate Single Table strategy.



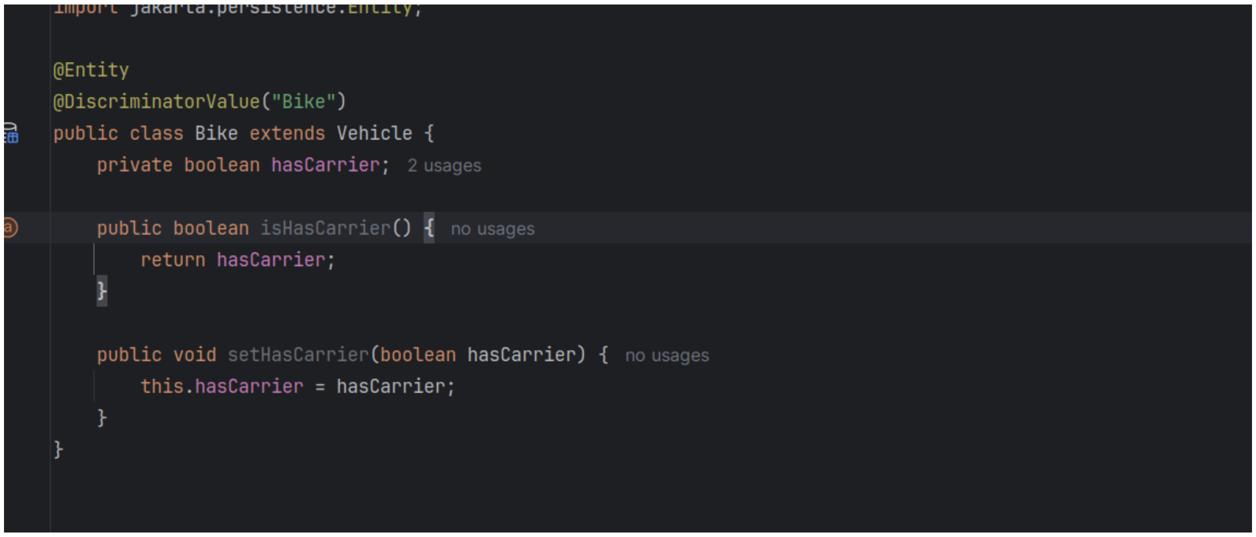
```
mysql> show tables;
+---------------------+
| Tables_in_SpringJpa |
+---------------------+
| employee_table      |
| vehicle             |
+---------------------+
2 rows in set (0.00 sec)
```



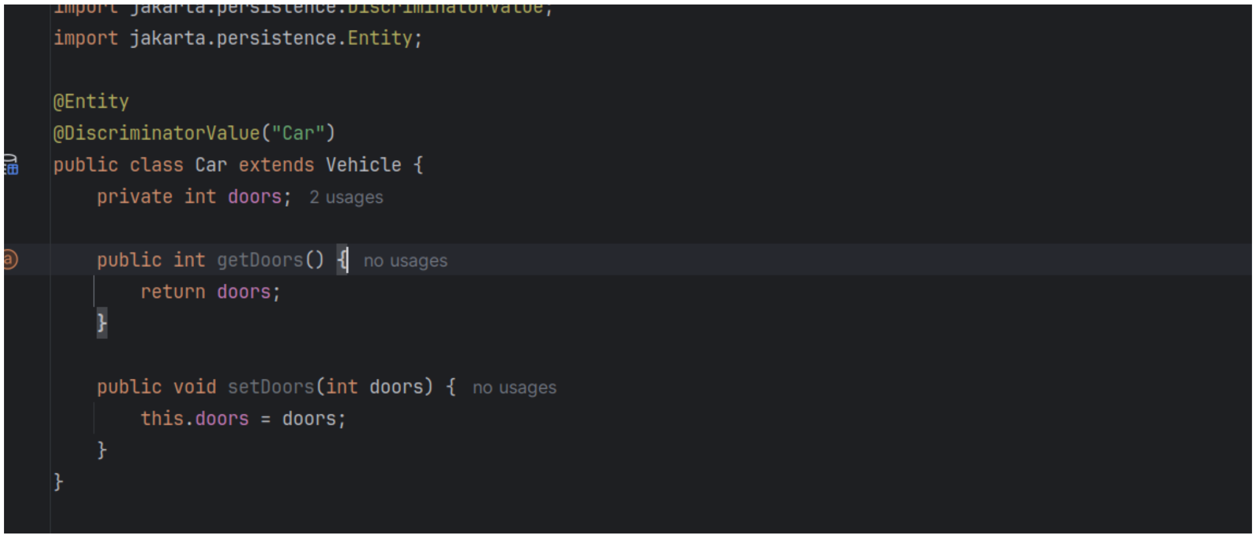
```
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| vehicle_type | varchar(31)  | NO   |     | NULL    |                |
| id           | bigint       | NO   | PRI | NULL    | auto_increment |
| tyres        | int          | NO   |     | NULL    |                |
| names        | varchar(255) | YES  |     | NULL    |                |
| has_carrier  | bit(1)       | YES  |     | NULL    |                |
| doors        | int          | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```



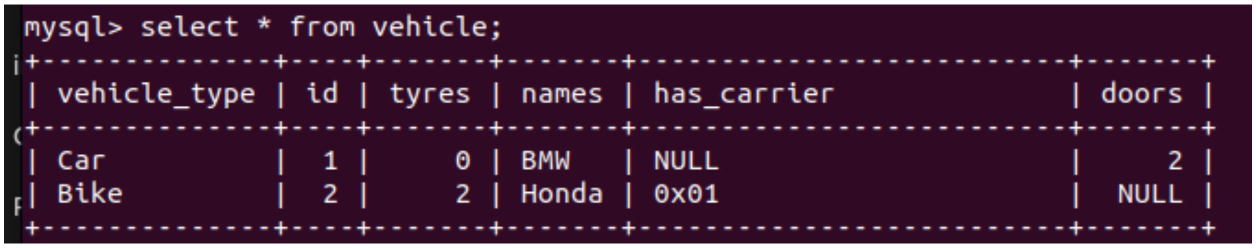
```
@Entity  2 inheritors
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="vehicle_type", discriminatorType = DiscriminatorType.STRING)
public abstract class Vehicle {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private int Tyres;  2 usages
    private String names;  2 usages
```



```java
import jakarta.persistence.Entity;

@Entity
@DiscriminatorValue("Bike")
public class Bike extends Vehicle {
    private boolean hasCarrier;

    public boolean isHasCarrier() {
        return hasCarrier;
    }

    public void setHasCarrier(boolean hasCarrier) {
        this.hasCarrier = hasCarrier;
    }
}
```

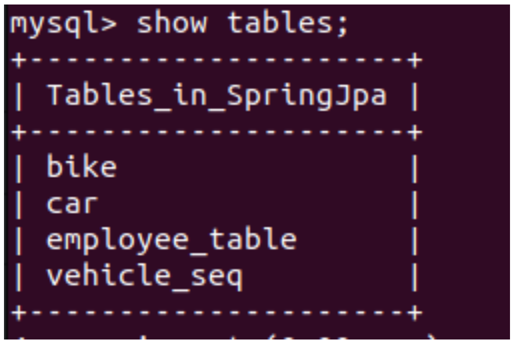


```java
import jakarta.persistence.DiscriminatorValue;
import jakarta.persistence.Entity;

@Entity
@DiscriminatorValue("Car")
public class Car extends Vehicle {
    private int doors;

    public int getDoors() {
        return doors;
    }

    public void setDoors(int doors) {
        this.doors = doors;
    }
}
```

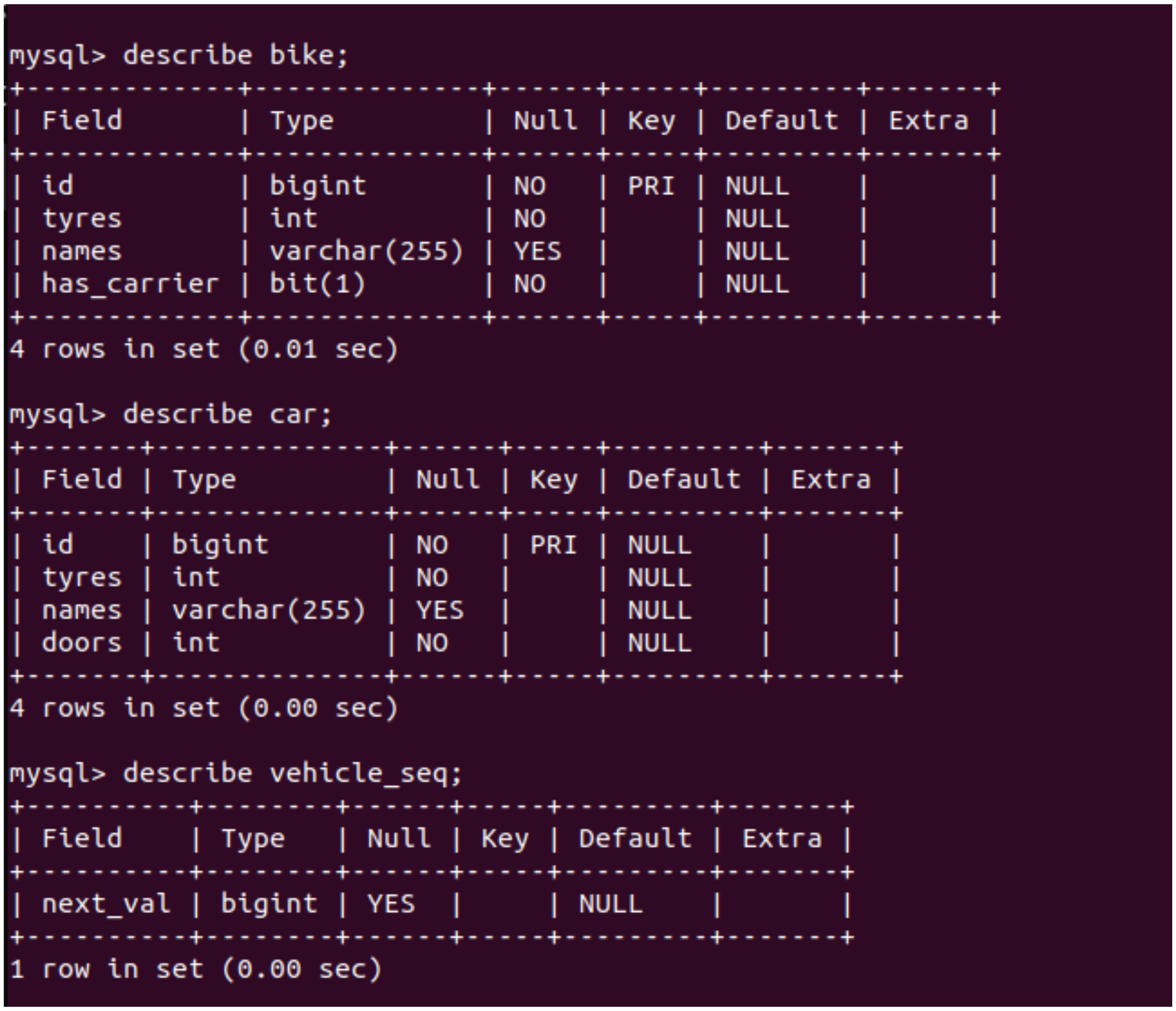


```
mysql> select * from vehicle;
+--------------+----+-------+-------+--------------------------+-------+
| vehicle_type | id | tyres | names | has_carrier              | doors |
+--------------+----+-------+-------+--------------------------+-------+
| Car          |  1 |     0 | BMW   | NULL                     |     2 |
| Bike         |  2 |     2 | Honda | 0x01                     |  NULL |
+--------------+----+-------+-------+--------------------------+-------+
```

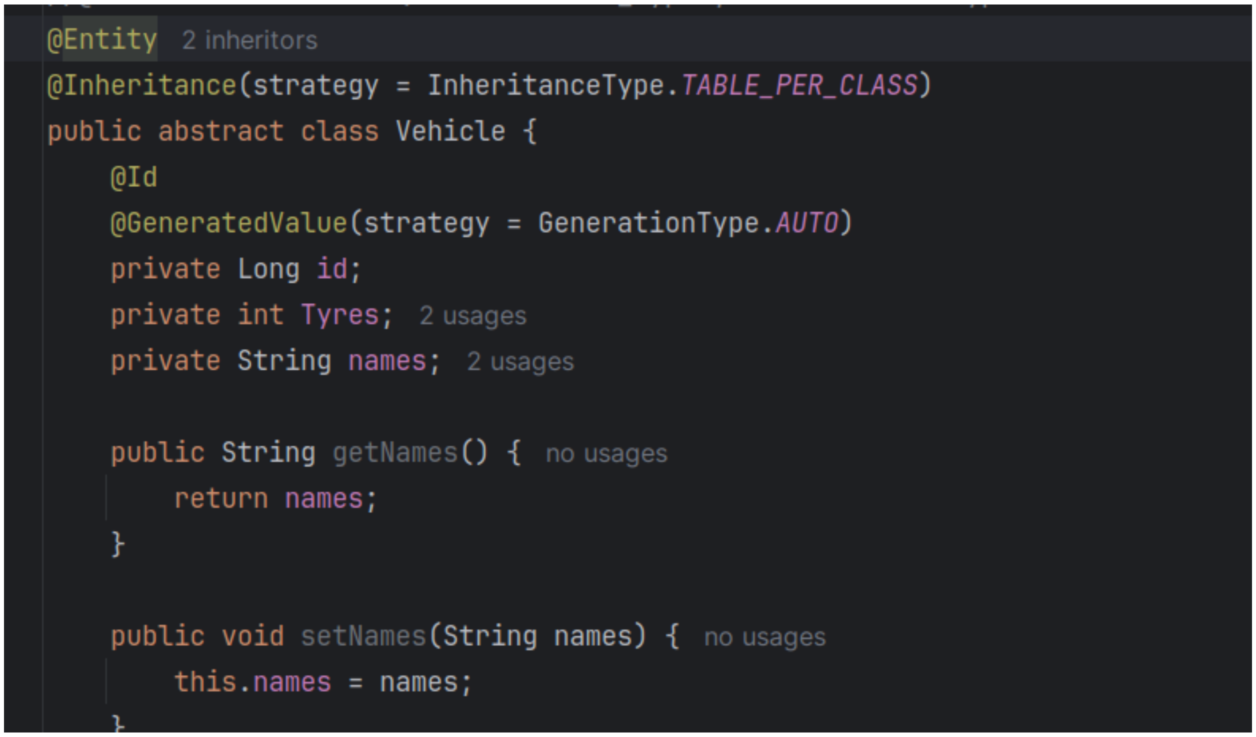
2) Implement and demonstrate Table Per Class strategy.



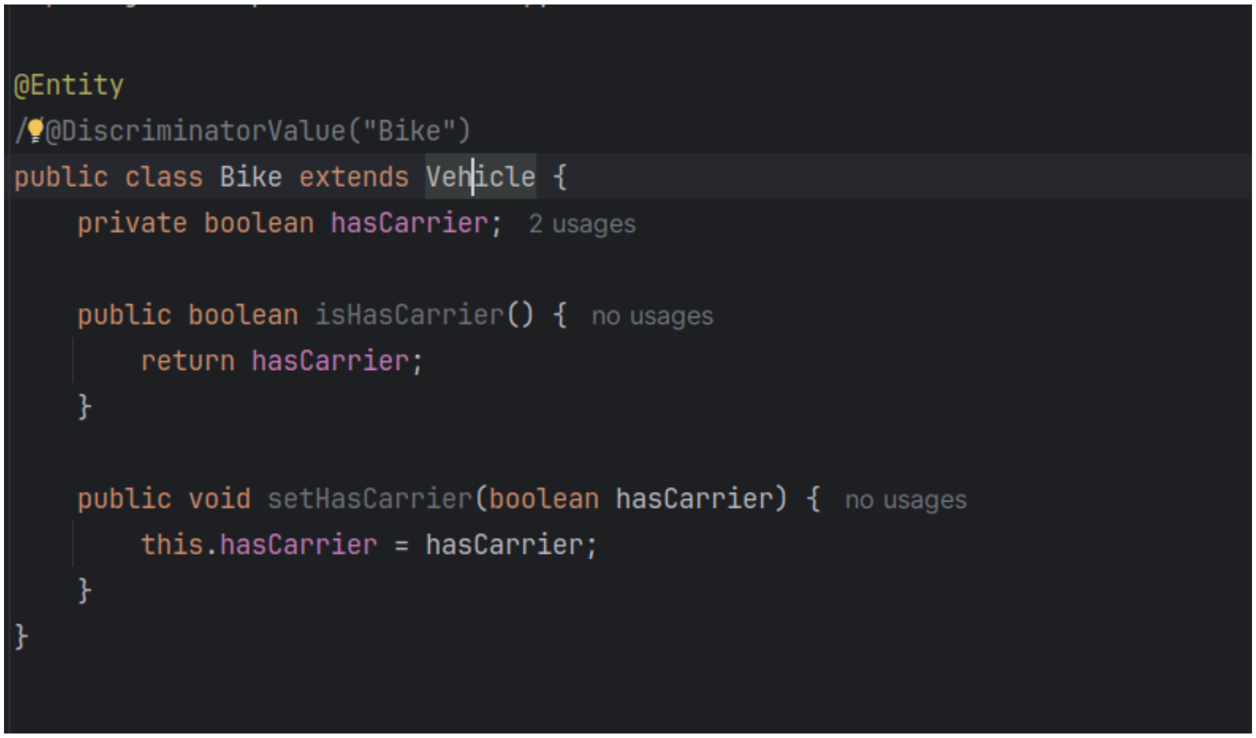
```
mysql> show tables;
+---------------------+
| Tables_in_SpringJpa |
+---------------------+
| bike                |
| car                 |
| employee_table      |
| vehicle_seq         |
+---------------------+
```



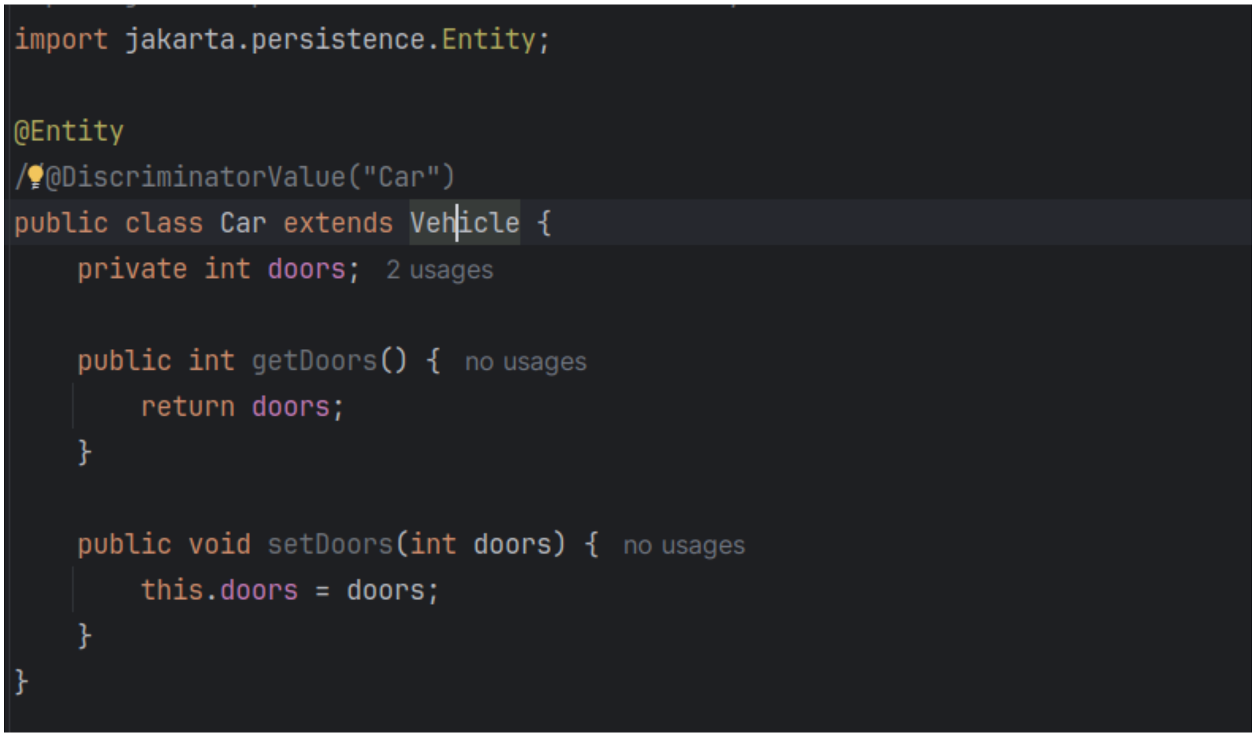
```
mysql> describe bike;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | bigint       | NO   | PRI | NULL    |       |
| tyres       | int          | NO   |     | NULL    |       |
| names       | varchar(255) | YES  |     | NULL    |       |
| has_carrier | bit(1)       | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> describe car;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | bigint       | NO   | PRI | NULL    |       |
| tyres | int          | NO   |     | NULL    |       |
| names | varchar(255) | YES  |     | NULL    |       |
| doors | int          | NO   |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> describe vehicle_seq;
+----------+--------+------+-----+---------+-------+
| Field    | Type   | Null | Key | Default | Extra |
+----------+--------+------+-----+---------+-------+
| next_val | bigint | YES  |     | NULL    |       |
+----------+--------+------+-----+---------+-------+
1 row in set (0.00 sec)
```



```java
@Entity  2 inheritors
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Vehicle {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private int Tyres;  2 usages
    private String names;  2 usages

    public String getNames() {  no usages
        return names;
    }

    public void setNames(String names) {  no usages
        this.names = names;
    }
```

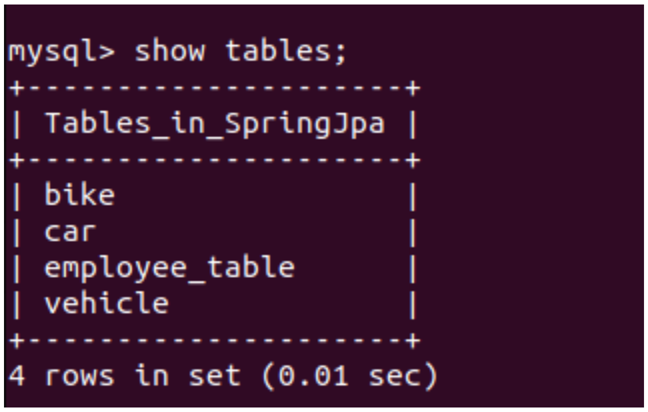


```java

@Entity
//@DiscriminatorValue("Bike")
public class Bike extends Vehicle {
    private boolean hasCarrier;  2 usages

    public boolean isHasCarrier() {  no usages
        return hasCarrier;
    }

    public void setHasCarrier(boolean hasCarrier) {  no usages
        this.hasCarrier = hasCarrier;
    }
}
```

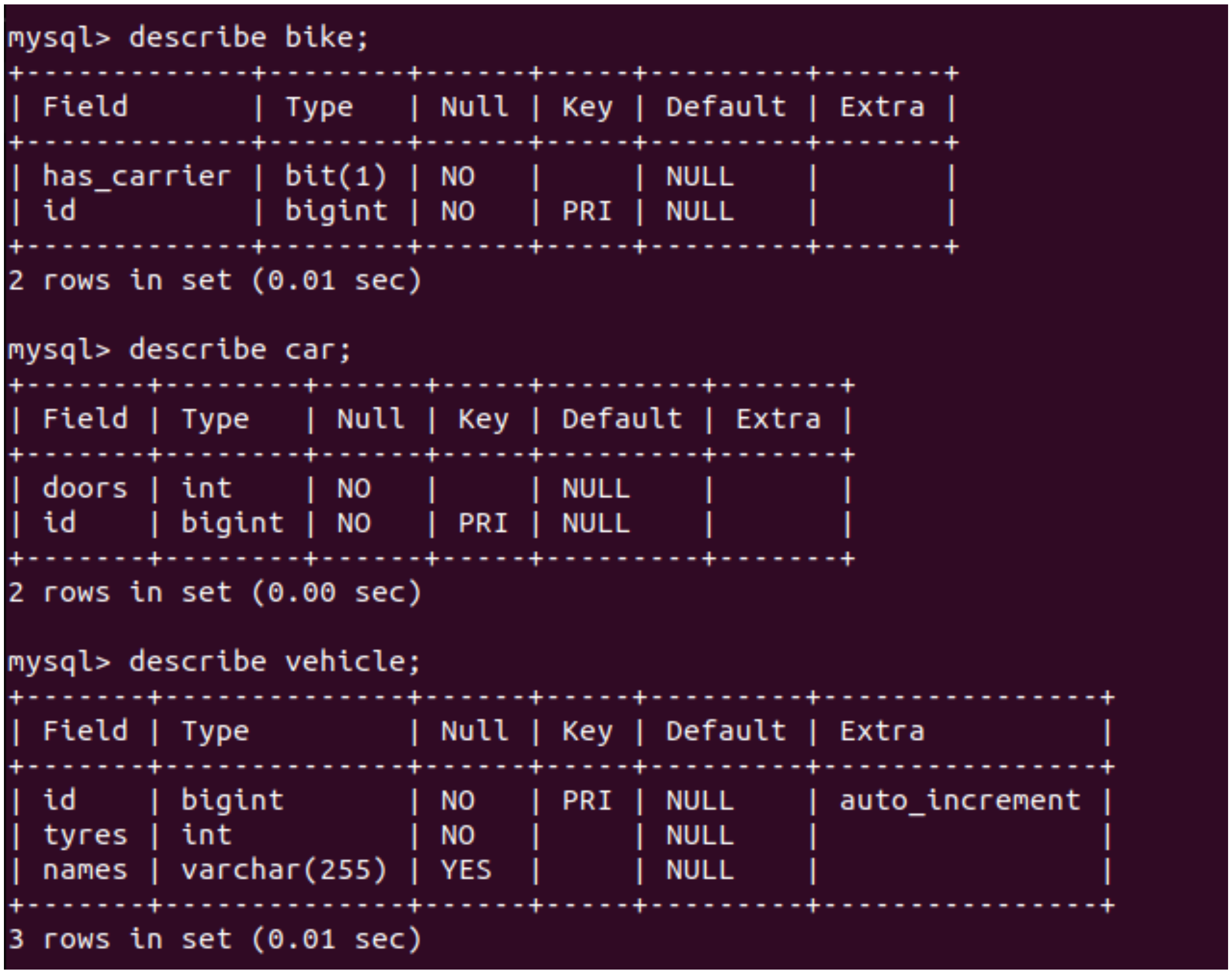


```java
import jakarta.persistence.Entity;

@Entity
//@DiscriminatorValue("Car")
public class Car extends Vehicle {
    private int doors;  2 usages

    public int getDoors() {  no usages
        return doors;
    }

    public void setDoors(int doors) {  no usages
        this.doors = doors;
    }
}
```

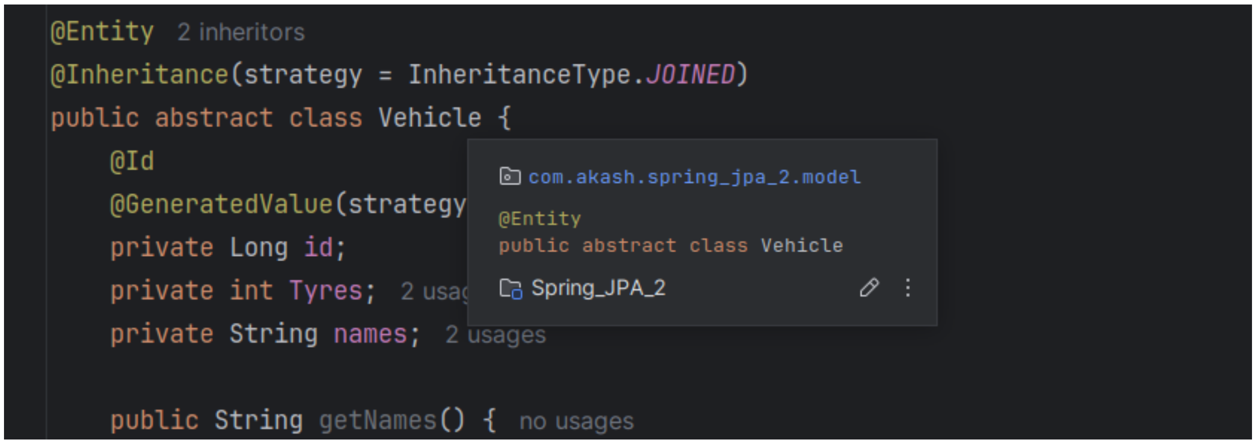
3) Implement and demonstrate Joined strategy.



```
mysql> show tables;
+---------------------+
| Tables_in_SpringJpa |
+---------------------+
| bike                |
| car                 |
| employee_table      |
| vehicle             |
+---------------------+
4 rows in set (0.01 sec)
```



```
mysql> describe bike;
+-------------+--------+------+-----+---------+-------+
| Field       | Type   | Null | Key | Default | Extra |
+-------------+--------+------+-----+---------+-------+
| has_carrier | bit(1) | NO   |     | NULL    |       |
| id          | bigint | NO   | PRI | NULL    |       |
+-------------+--------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> describe car;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| doors | int    | NO   |     | NULL    |       |
| id    | bigint | NO   | PRI | NULL    |       |
+-------+--------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> describe vehicle;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | bigint       | NO   | PRI | NULL    | auto_increment |
| tyres | int          | NO   |     | NULL    |                |
| names | varchar(255) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

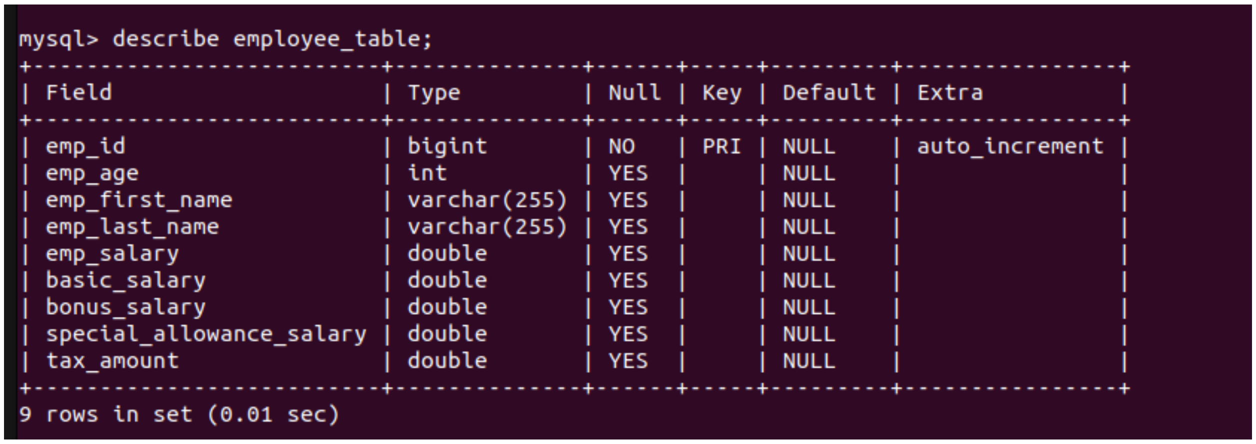


```java
@Entity  2 inheritors
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Vehicle {
    @Id
    @GeneratedValue(strategy
    private Long id;
    private int Tyres;  2 usag
    private String names;  2 usages

    public String getNames() {  no usages
```

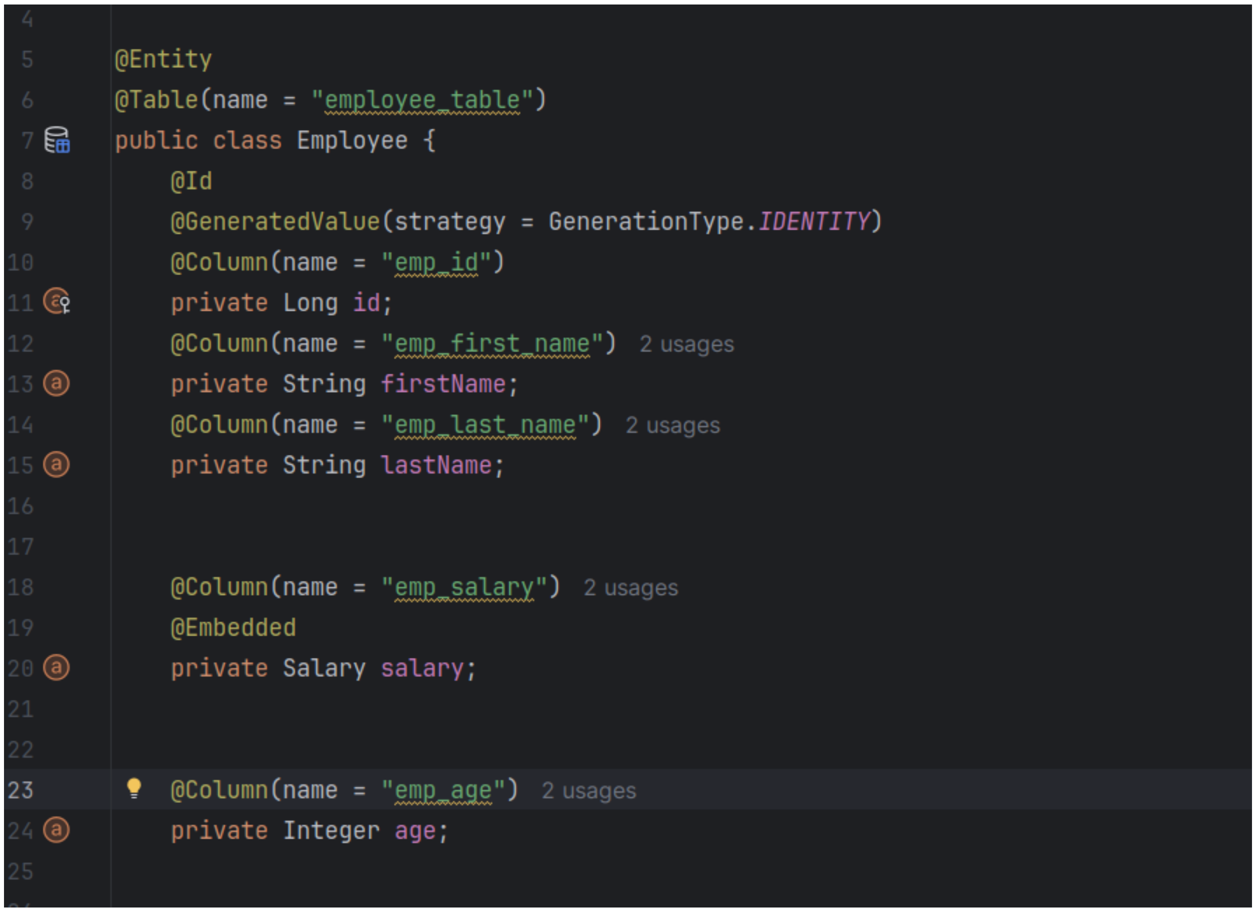
```
com.akash.spring_jpa_2.model
@Entity
public abstract class Vehicle
Spring_JPA_2
```

Q4)Component Mapping:

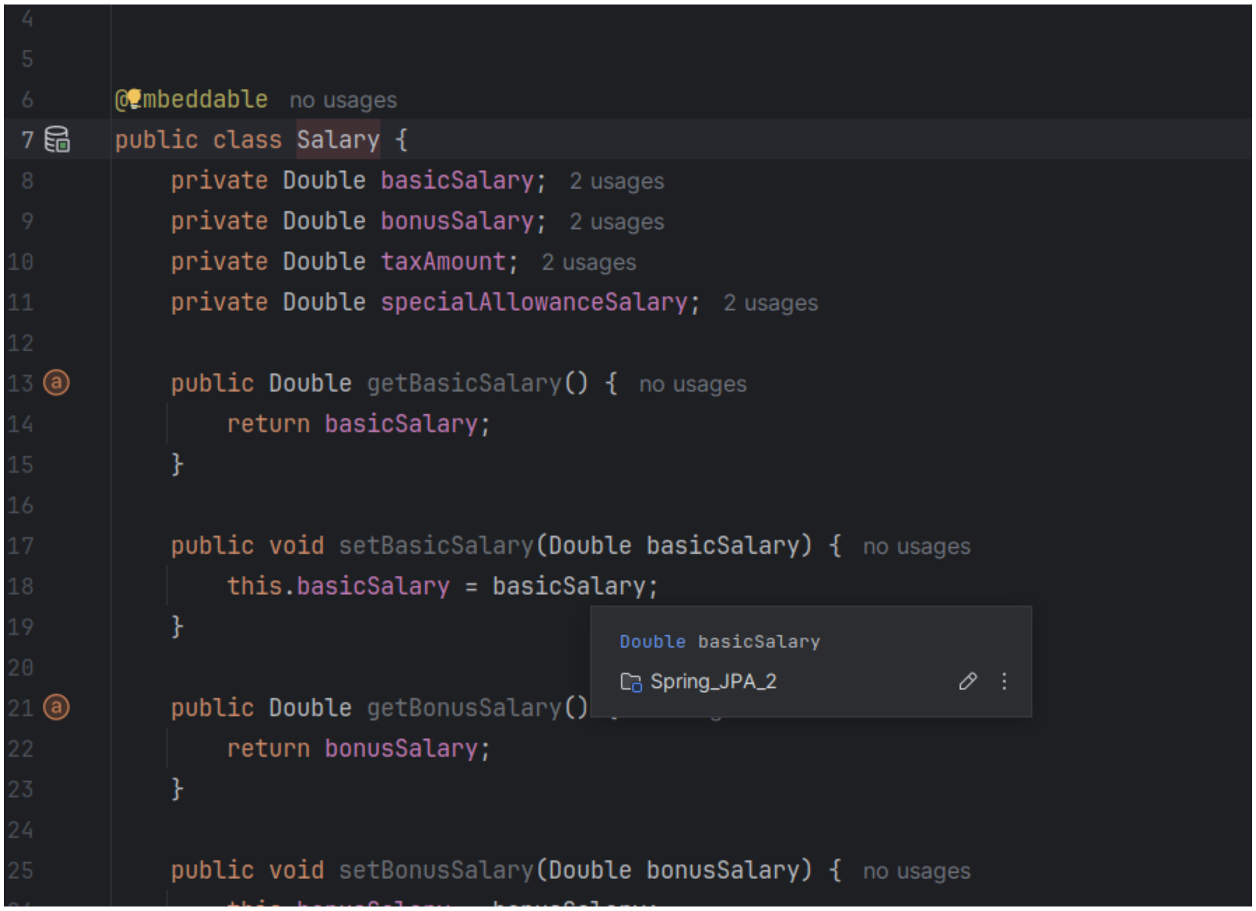
1) Implement and demonstrate Embedded mapping using employee table having following fields: id, firstName, lastName, age, basicSalary, bonusSalary, taxAmount, specialAllowanceSalary.



```
mysql> describe employee_table;
+-------------------------+--------------+------+-----+---------+----------------+
| Field                   | Type         | Null | Key | Default | Extra          |
+-------------------------+--------------+------+-----+---------+----------------+
| emp_id                  | bigint       | NO   | PRI | NULL    | auto_increment |
| emp_age                 | int          | YES  |     | NULL    |                |
| emp_first_name          | varchar(255) | YES  |     | NULL    |                |
| emp_last_name           | varchar(255) | YES  |     | NULL    |                |
| emp_salary              | double       | YES  |     | NULL    |                |
| basic_salary            | double       | YES  |     | NULL    |                |
| bonus_salary            | double       | YES  |     | NULL    |                |
| special_allowance_salary | double      | YES  |     | NULL    |                |
| tax_amount              | double       | YES  |     | NULL    |                |
+-------------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.01 sec)
```



```java
@Entity
@Table(name = "employee_table")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "emp_id")
    private Long id;
    @Column(name = "emp_first_name")  2 usages
    private String firstName;
    @Column(name = "emp_last_name")  2 usages
    private String lastName;


    @Column(name = "emp_salary")  2 usages
    @Embedded
    private Salary salary;


    @Column(name = "emp_age")  2 usages
    private Integer age;

```



```
@Embeddable  no usages
public class Salary {
    private Double basicSalary;  2 usages
    private Double bonusSalary;  2 usages
    private Double taxAmount;  2 usages
    private Double specialAllowanceSalary;  2 usages

    public Double getBasicSalary() {  no usages
        return basicSalary;
    }

    public void setBasicSalary(Double basicSalary) {  no usages
        this.basicSalary = basicSalary;
    }

    public Double getBonusSalary()
        return bonusSalary;
    }

    public void setBonusSalary(Double bonusSalary) {  no usages
```