Spring Data JPA part 1 Assignment

(1) Create an Employee Entity which contains following fields

Name

Id

Age

Location

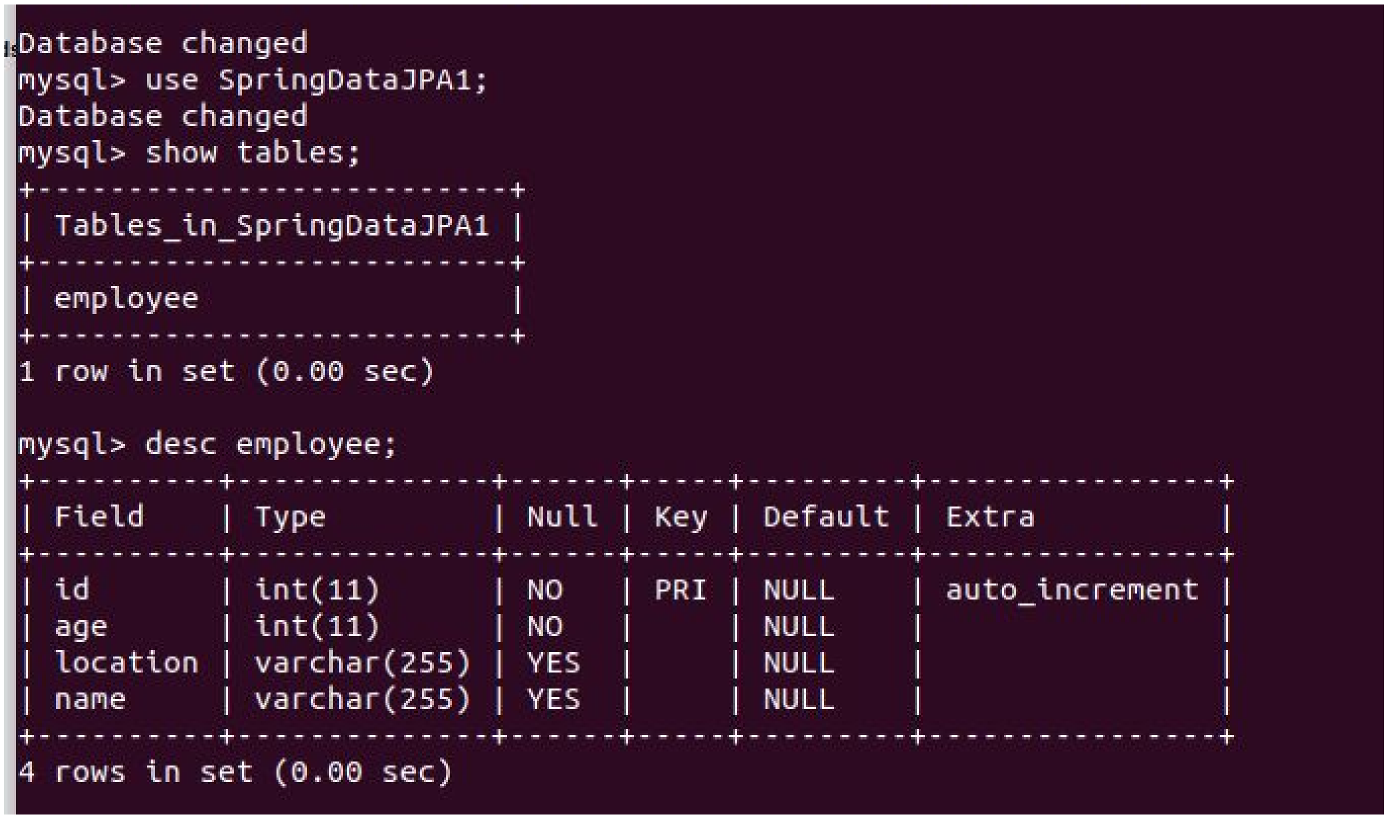
CODE

Employee.java

```
@Entity
public class Employee {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name;
  private int age;
  private String location;

  public Employee() {
  }

  public int getId() {
     return id;
  }

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

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }
```

```
    public int getAge() {
        return age;
    }

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

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }
}
```

<u>OUTPUT</u>



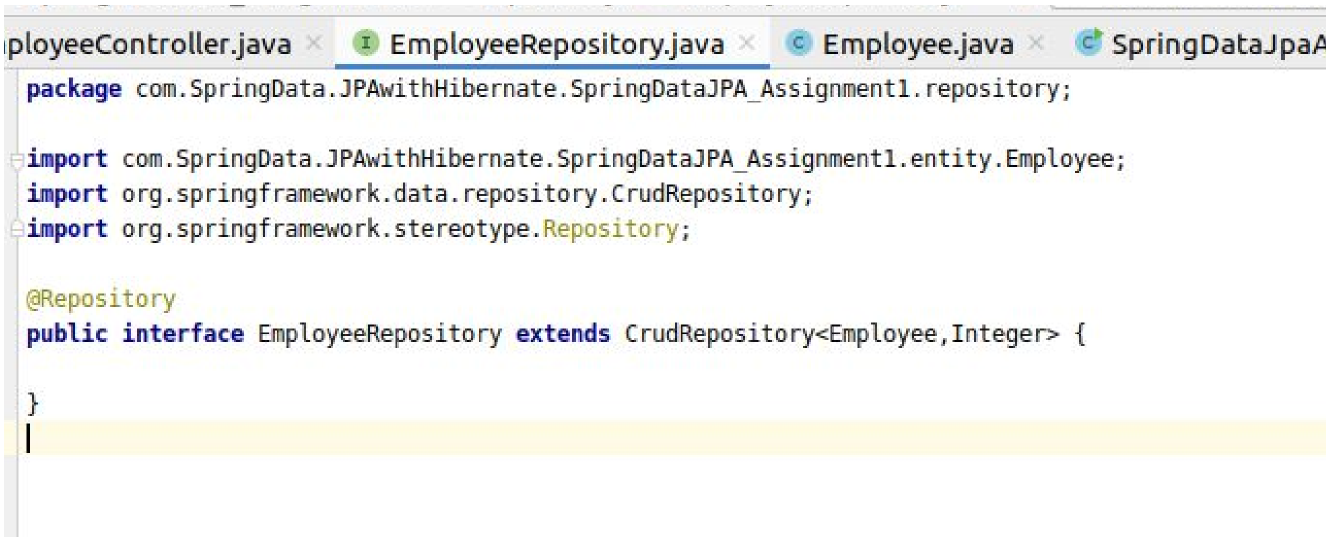
```
Database changed
mysql> use SpringDataJPA1;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_SpringDataJPA1 |
+--------------------------+
| employee                 |
+--------------------------+
1 row in set (0.00 sec)

mysql> desc employee;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int(11)      | NO   | PRI | NULL    | auto_increment |
| age      | int(11)      | NO   |     | NULL    |                |
| location | varchar(255) | YES  |     | NULL    |                |
| name     | varchar(255) | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

(2) Set up EmployeeRepository with Spring Data JPA

CODE

```
@Repository
public interface EmployeeRepository extends CrudRepository<Employee,Integer> {

}
```

OUTPUT



ployeeController.java | EmployeeRepository.java | Employee.java | SpringDataJpaA

```
package com.SpringData.JPAwithHibernate.SpringDataJPA_Assignment1.repository;

import com.SpringData.JPAwithHibernate.SpringDataJPA_Assignment1.entity.Employee;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeRepository extends CrudRepository<Employee,Integer> {

}
```

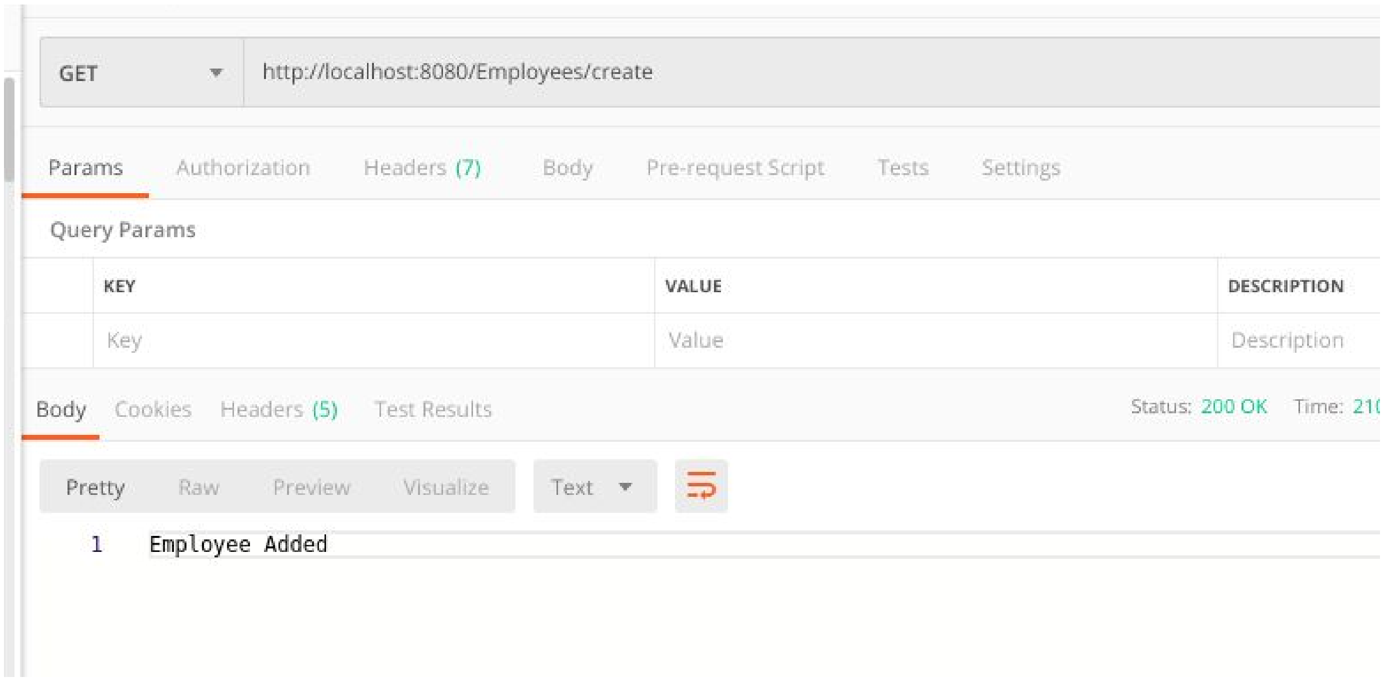
(3) Perform Create Operation on Entity using Spring Data JPA

CODE

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

  @Autowired
  EmployeeRepository employeeRepository;

  @GetMapping("/create")
  public String addEmployee(){
     Employee employee = new Employee();
     employee.setName("Aayushi");
     employee.setAge(24);
     employee.setLocation("Delhi");
     employeeRepository.save(employee);
     return "Employee Added";
  }
```

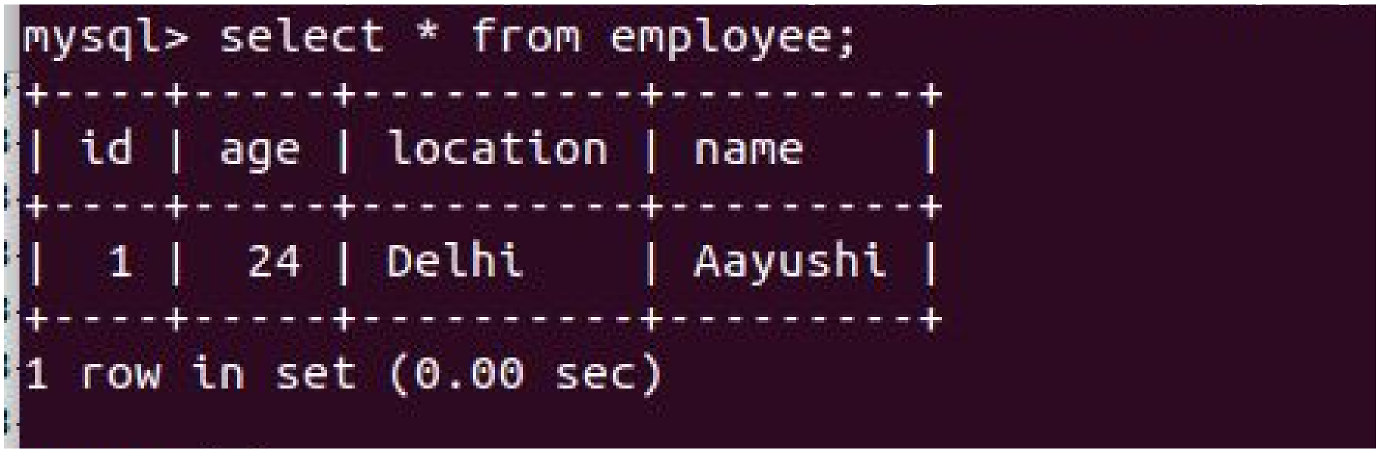
}

OUTPUT



```
http://localhost:8080/Employees/create
```

```
Employee Added
```



```
mysql> select * from employee;
+----+-----+----------+---------+
| id | age | location | name    |
+----+-----+----------+---------+
|  1 |  24 | Delhi    | Aayushi |
+----+-----+----------+---------+
1 row in set (0.00 sec)
```

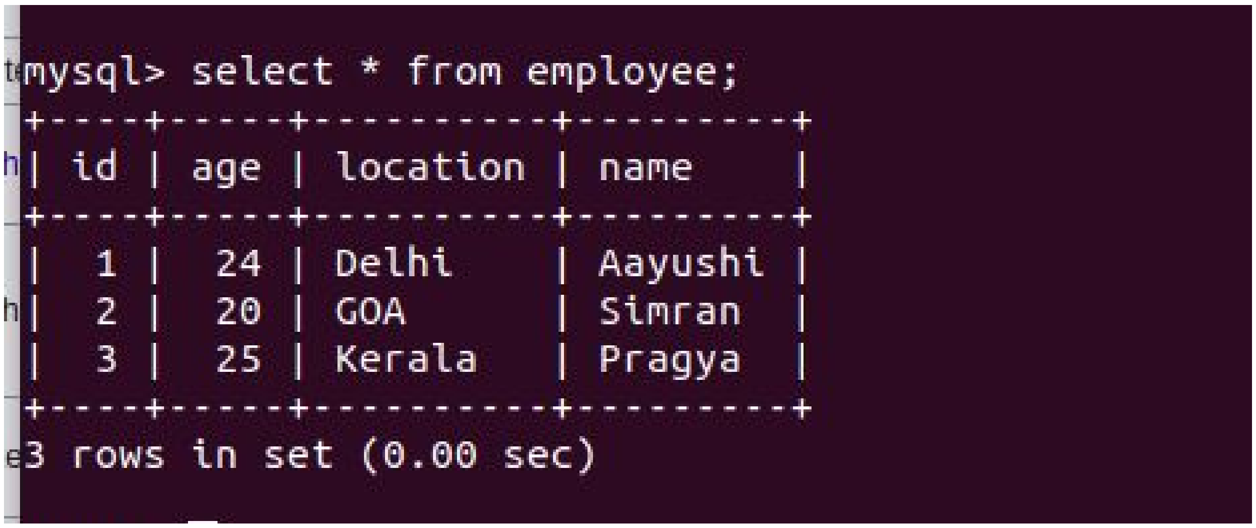
(4) Perform Update Operation on Entity using Spring Data JPA

CODE

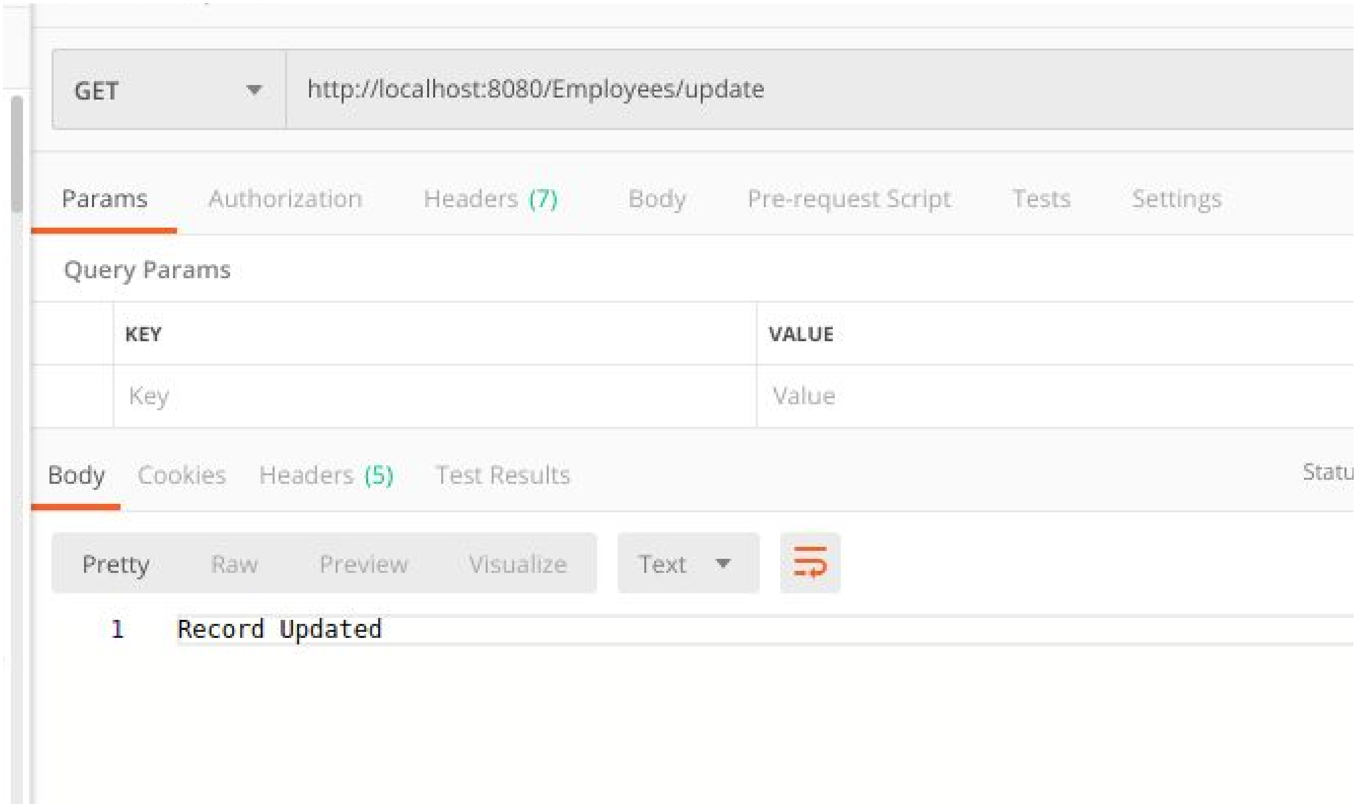
```
//UPDATE OPERATION
@GetMapping("/update")
```

```
public String updateEmployee(){
  Optional<Employee> employee = employeeRepository.findById(3);
  if (employee.isPresent())
  {
     Employee employee1 = employee.get();
     employee1.setLocation("Chandigarh");
     employee1.setAge(22);
     employeeRepository.save(employee1);
     return "Record Updated";
  }
  else
     return "Record Not found";
}
```

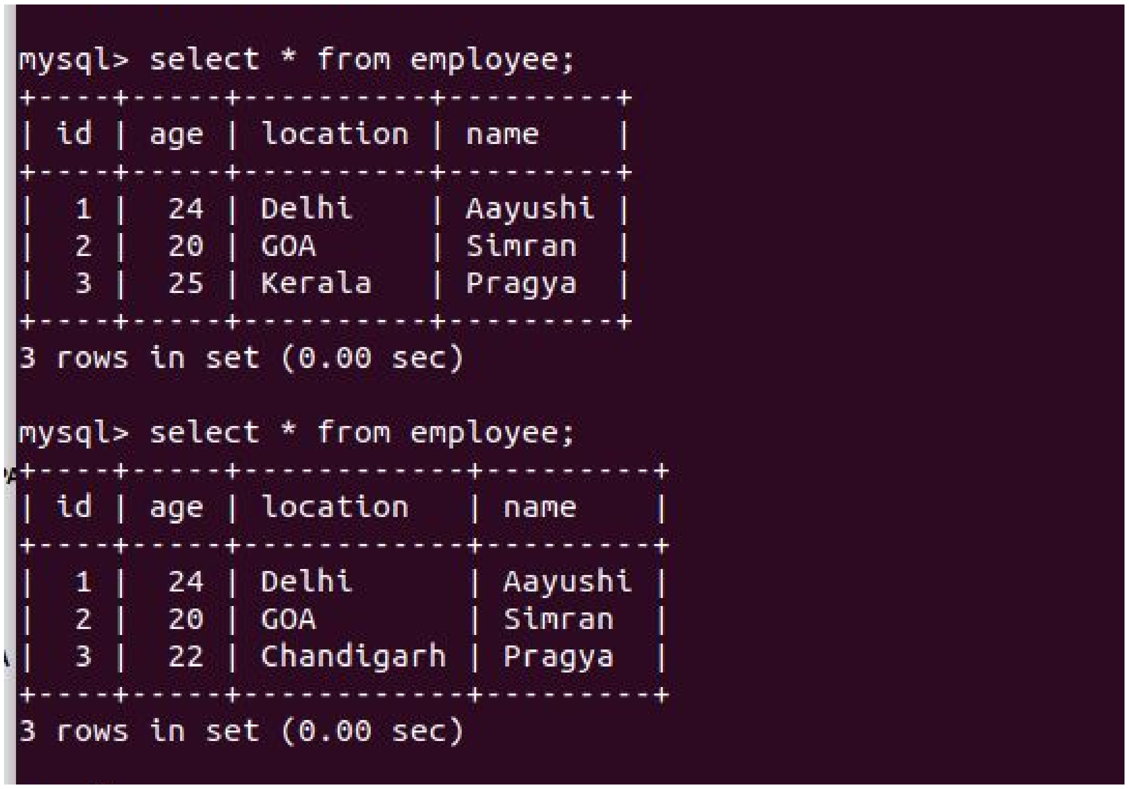
OUTPUT



```
mysql> select * from employee;
+----+-----+----------+---------+
| id | age | location | name    |
+----+-----+----------+---------+
|  1 |  24 | Delhi    | Aayushi |
|  2 |  20 | GOA      | Simran  |
|  3 |  25 | Kerala   | Pragya  |
+----+-----+----------+---------+
3 rows in set (0.00 sec)
```



```
1    Record Updated
```



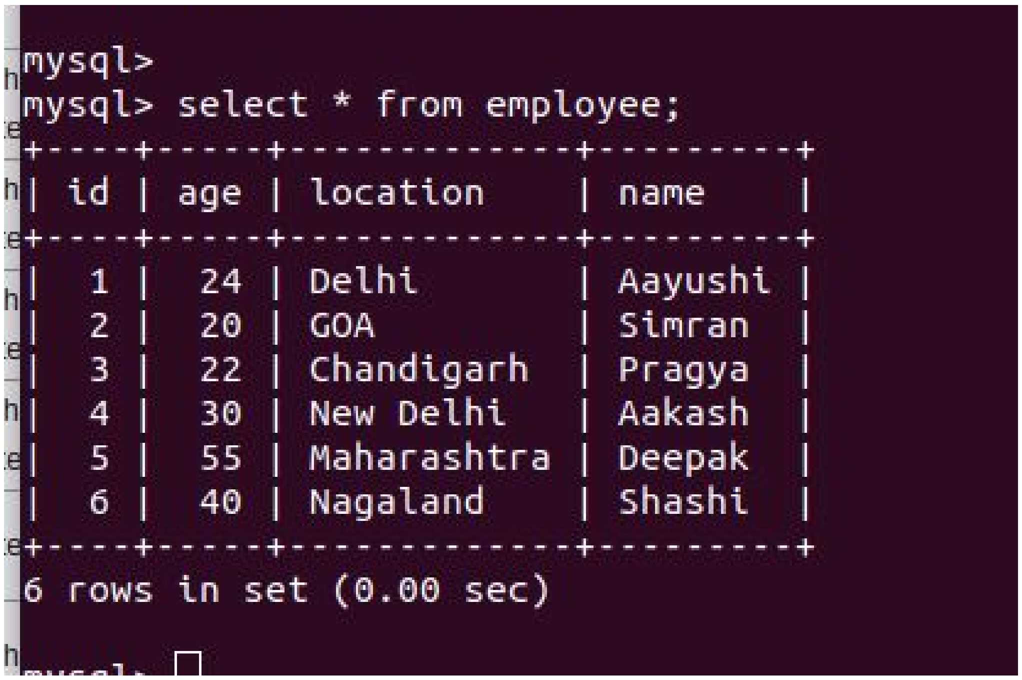
```
mysql> select * from employee;
+----+-----+----------+---------+
| id | age | location | name    |
+----+-----+----------+---------+
|  1 |  24 | Delhi    | Aayushi |
|  2 |  20 | GOA      | Simran  |
|  3 |  25 | Kerala   | Pragya  |
+----+-----+----------+---------+
3 rows in set (0.00 sec)

mysql> select * from employee;
+----+-----+------------+---------+
| id | age | location   | name    |
+----+-----+------------+---------+
|  1 |  24 | Delhi      | Aayushi |
|  2 |  20 | GOA        | Simran  |
|  3 |  22 | Chandigarh | Pragya  |
+----+-----+------------+---------+
3 rows in set (0.00 sec)
```

(5) Perform Delete Operation on Entity using Spring Data JPA

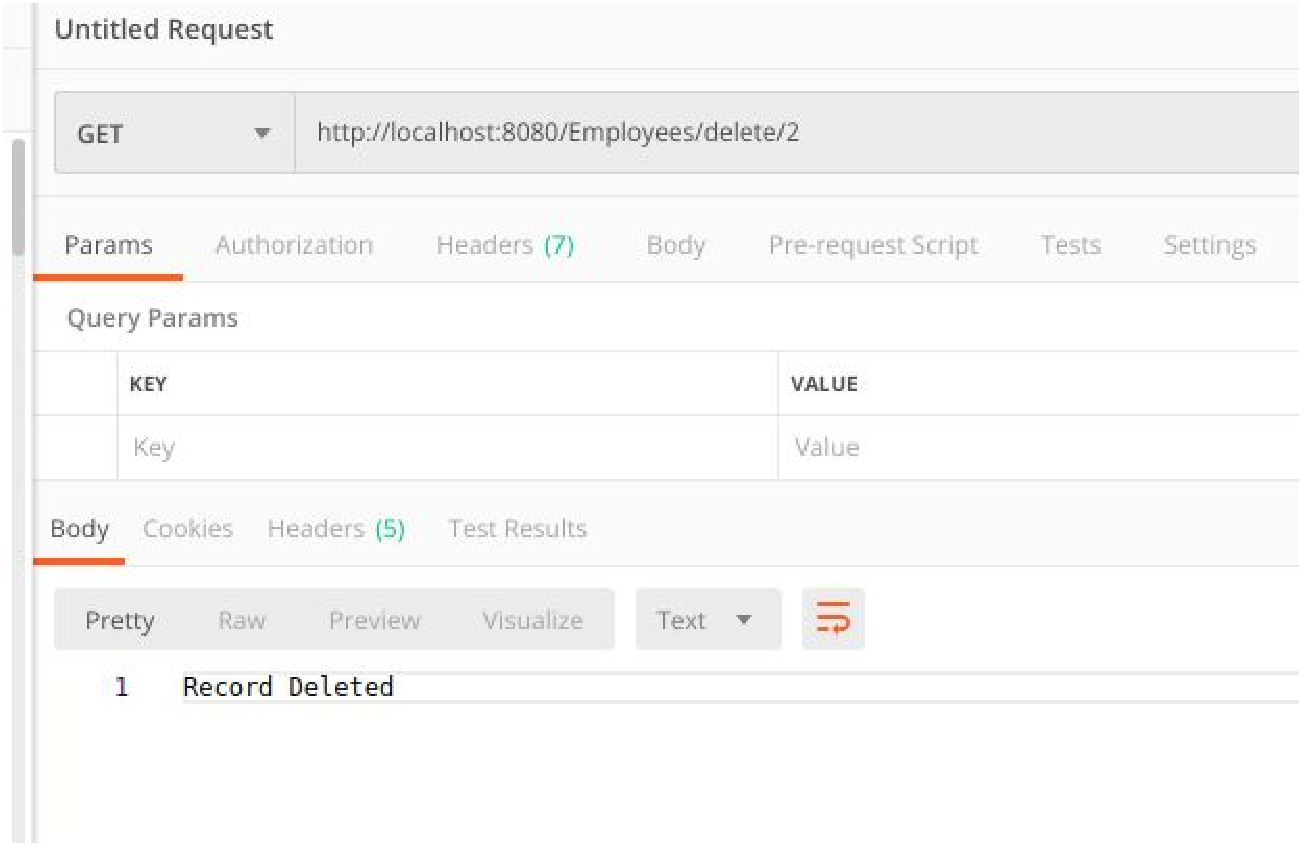
CODE

```
//DELETE OPERATION
@GetMapping("/delete/{id}")
public String deleteEmployee(@PathVariable Integer id){
   Optional<Employee> employee = employeeRepository.findById(id);
   if (employee.isPresent())
   {
       employeeRepository.deleteById(id);
       return "Record Deleted";
   }
   else
       return "Record Not found";
}
```

OUTPUT

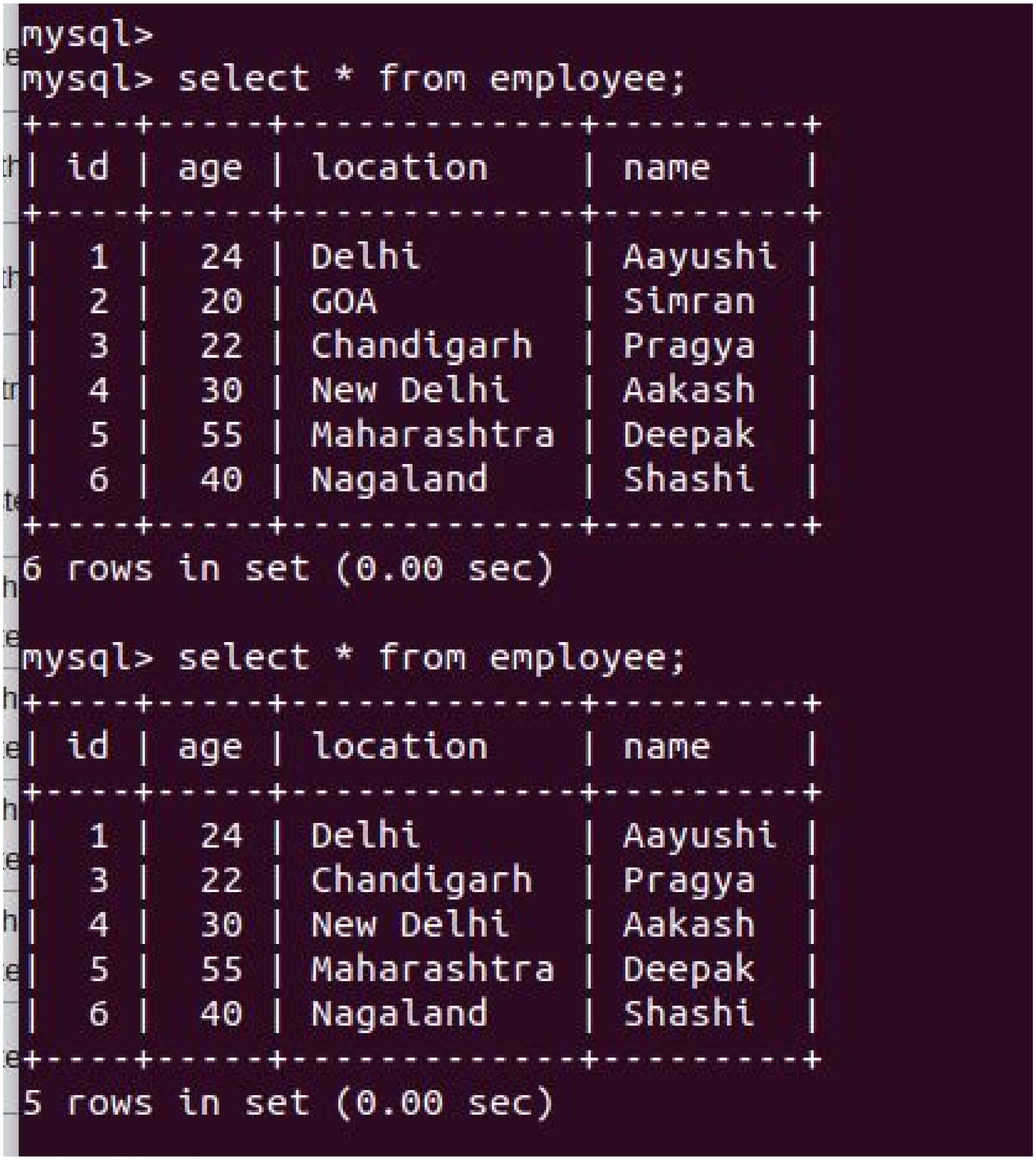


```
mysql>
mysql> select * from employee;
+----+-----+-------------+---------+
| id | age | location    | name    |
+----+-----+-------------+---------+
|  1 |  24 | Delhi       | Aayushi |
|  2 |  20 | GOA         | Simran  |
|  3 |  22 | Chandigarh  | Pragya  |
|  4 |  30 | New Delhi   | Aakash  |
|  5 |  55 | Maharashtra | Deepak  |
|  6 |  40 | Nagaland    | Shashi  |
+----+-----+-------------+---------+
6 rows in set (0.00 sec)

mysql>
```


Untitled Request

GET http://localhost:8080/Employees/delete/2

```
Record Deleted
```

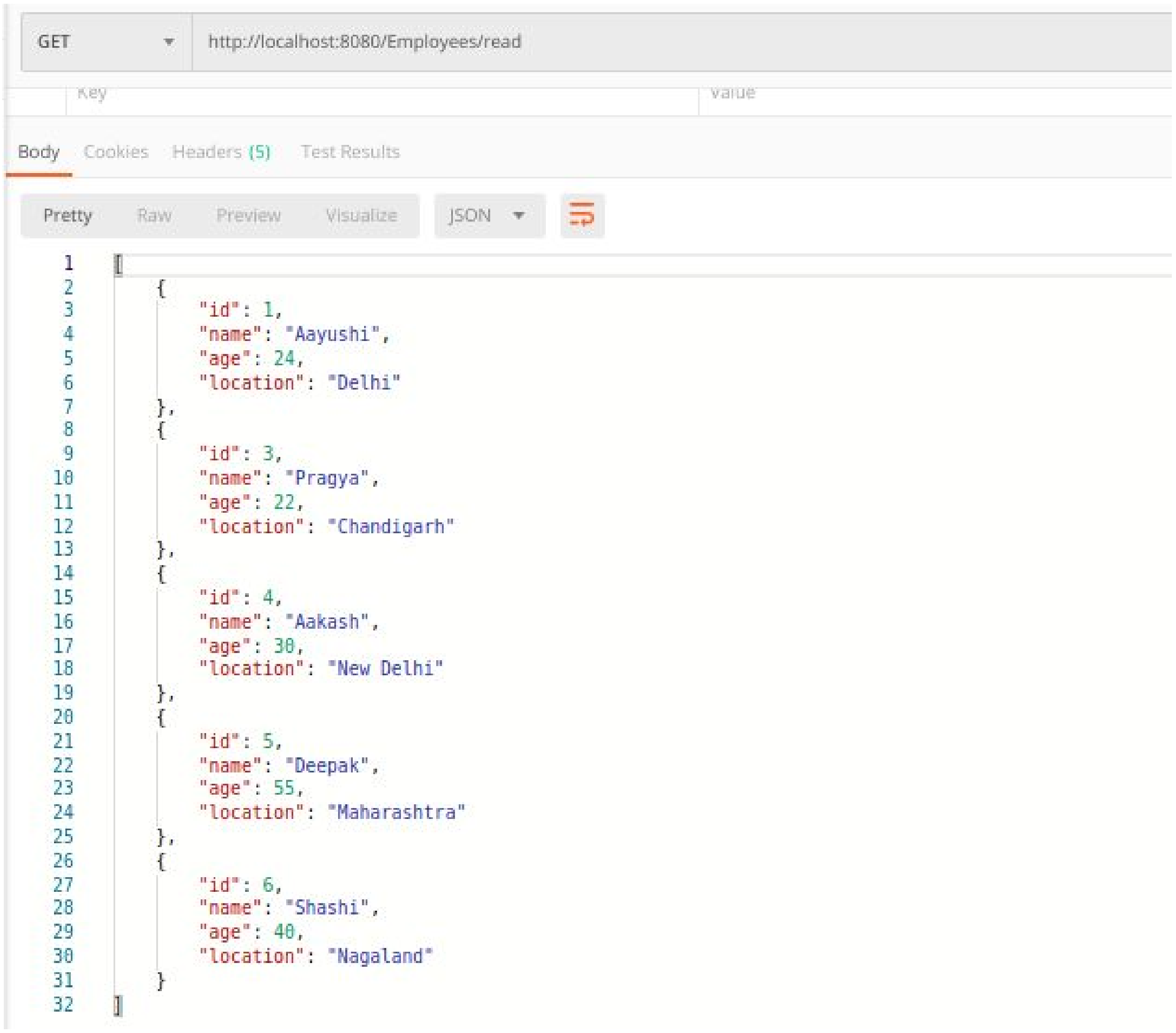

```
mysql>
mysql> select * from employee;
+----+-----+-------------+---------+
| id | age | location    | name    |
+----+-----+-------------+---------+
|  1 |  24 | Delhi       | Aayushi |
|  2 |  20 | GOA         | Simran  |
|  3 |  22 | Chandigarh  | Pragya  |
|  4 |  30 | New Delhi   | Aakash  |
|  5 |  55 | Maharashtra | Deepak  |
|  6 |  40 | Nagaland    | Shashi  |
+----+-----+-------------+---------+
6 rows in set (0.00 sec)

mysql> select * from employee;
+----+-----+-------------+---------+
| id | age | location    | name    |
+----+-----+-------------+---------+
|  1 |  24 | Delhi       | Aayushi |
|  3 |  22 | Chandigarh  | Pragya  |
|  4 |  30 | New Delhi   | Aakash  |
|  5 |  55 | Maharashtra | Deepak  |
|  6 |  40 | Nagaland    | Shashi  |
+----+-----+-------------+---------+
5 rows in set (0.00 sec)
```

(5) Perform Read Operation on Entity using Spring Data JPA

CODE

```
//READ OPERATION
@GetMapping("/read")
public List<Employee> readEmployee(){
  List<Employee> employeeList = (List<Employee>) employeeRepository.findAll();
  return employeeList;
}
```

OUTPUT



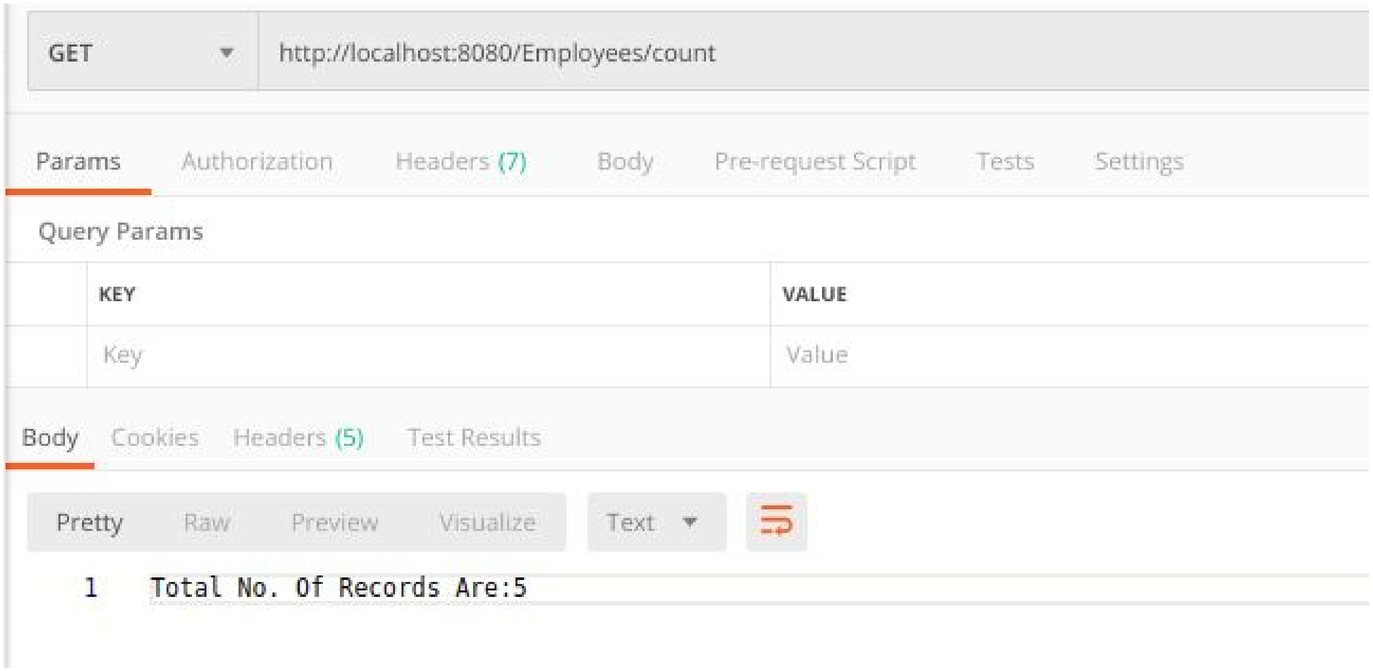
```
[
    {
        "id": 1,
        "name": "Aayushi",
        "age": 24,
        "location": "Delhi"
    },
    {
        "id": 3,
        "name": "Pragya",
        "age": 22,
        "location": "Chandigarh"
    },
    {
        "id": 4,
        "name": "Aakash",
        "age": 30,
        "location": "New Delhi"
    },
    {
        "id": 5,
        "name": "Deepak",
        "age": 55,
        "location": "Maharashtra"
    },
    {
        "id": 6,
        "name": "Shashi",
        "age": 40,
        "location": "Nagaland"
    }
]
```

(6) Get the total count of the number of Employees.

CODE

```
@GetMapping("/count")
public String countEmployee(){
  Long count = employeeRepository.count();
  return "Total No. Of Records Are:" +count;
}
```

OUTPUT



http://localhost:8080/Employees/count

```
Total No. Of Records Are:5
```

(7) Implement Pagination and Sorting on the bases of Employee Age.

CODE

EmployeeRepository.java

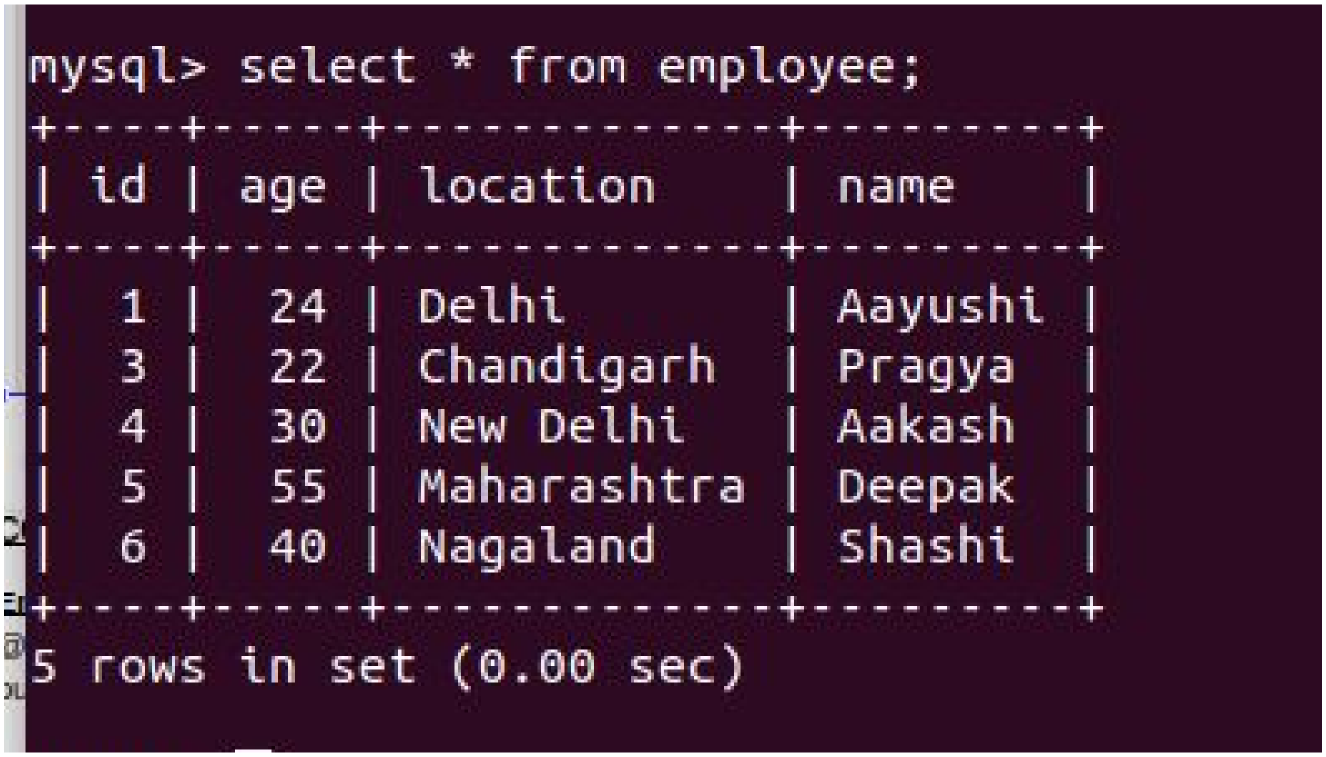
```
@Repository
public interface EmployeeRepository extends CrudRepository<Employee,Integer> {

  List<Employee> findByName(String name);

  List<Employee> findByNameLike(String desc);

  List<Employee> findByAgeBetween(Integer age1,Integer age2);
```

```
   List<Employee> findAll(Pageable pageable);
}
```

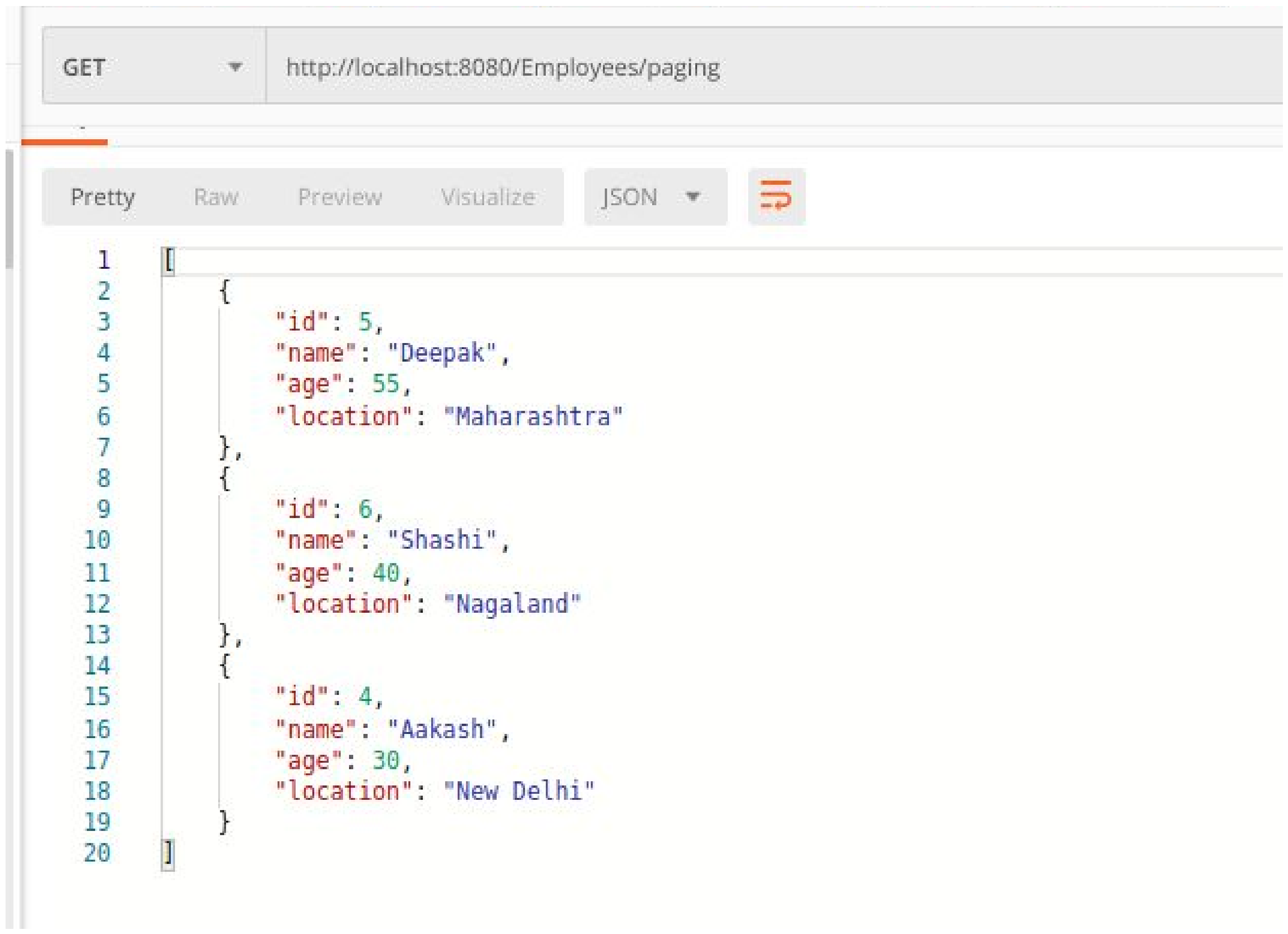
EmployeeController.java

```
   //PAGING AND SORTING
@GetMapping("/paging")
public List<Employee> employeeAgePagable(){
   Pageable pageable = PageRequest.of(0,3, Sort.Direction.DESC,"age");
   List<Employee> employeeList = (List<Employee>) employeeRepository.findAll(pageable);
   return employeeList;
}
```

OUTPUT



```
mysql> select * from employee;
+----+-----+-------------+---------+
| id | age | location    | name    |
+----+-----+-------------+---------+
|  1 |  24 | Delhi       | Aayushi |
|  3 |  22 | Chandigarh  | Pragya  |
|  4 |  30 | New Delhi   | Aakash  |
|  5 |  55 | Maharashtra | Deepak  |
|  6 |  40 | Nagaland    | Shashi  |
+----+-----+-------------+---------+
5 rows in set (0.00 sec)
```



http://localhost:8080/Employees/paging

```
[
    {
        "id": 5,
        "name": "Deepak",
        "age": 55,
        "location": "Maharashtra"
    },
    {
        "id": 6,
        "name": "Shashi",
        "age": 40,
        "location": "Nagaland"
    },
    {
        "id": 4,
        "name": "Aakash",
        "age": 30,
        "location": "New Delhi"
    }
]
```

(8) Create and use finder to find Employee by Name.

CODE

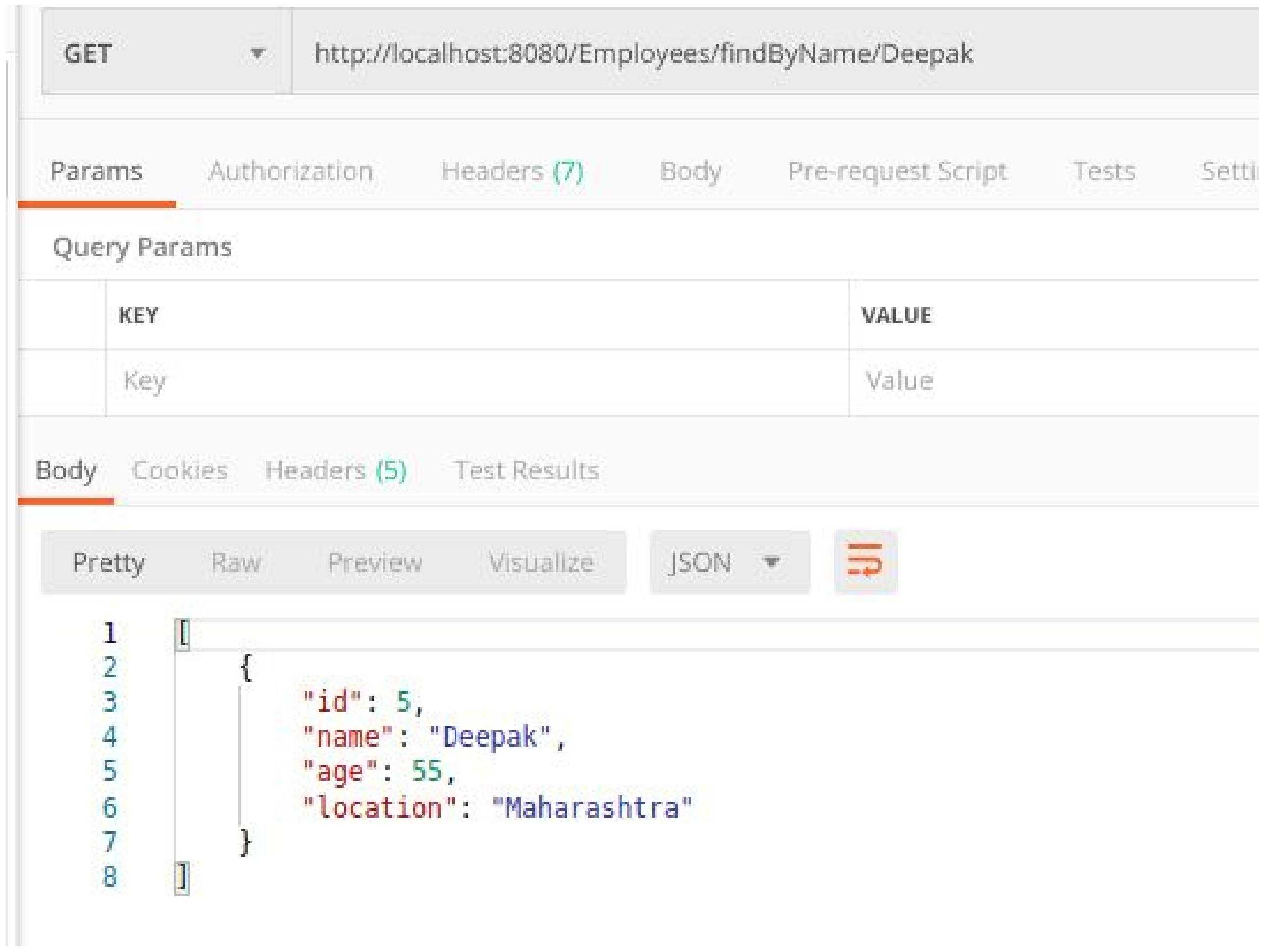
EmployeeRepository.java

```
@Repository
public interface EmployeeRepository extends CrudRepository<Employee,Integer> {

   List<Employee> findByName(String name);
}
```

EmployeeController.java

```
//FIND BY NAME OPERATION
@GetMapping("/findByName/{name}")
public List<Employee> findEmployeeByName(@PathVariable String name){
   List<Employee> employeeList = (List<Employee>) employeeRepository.findByName(name);
   return employeeList;
}
```

OUTPUT


GET http://localhost:8080/Employees/findByName/Deepak

```
[
    {
        "id": 5,
        "name": "Deepak",
        "age": 55,
        "location": "Maharashtra"
    }
]
```

(9) Create and use finder to find Employees starting with A character.

CODE

EmployeeRepository.java

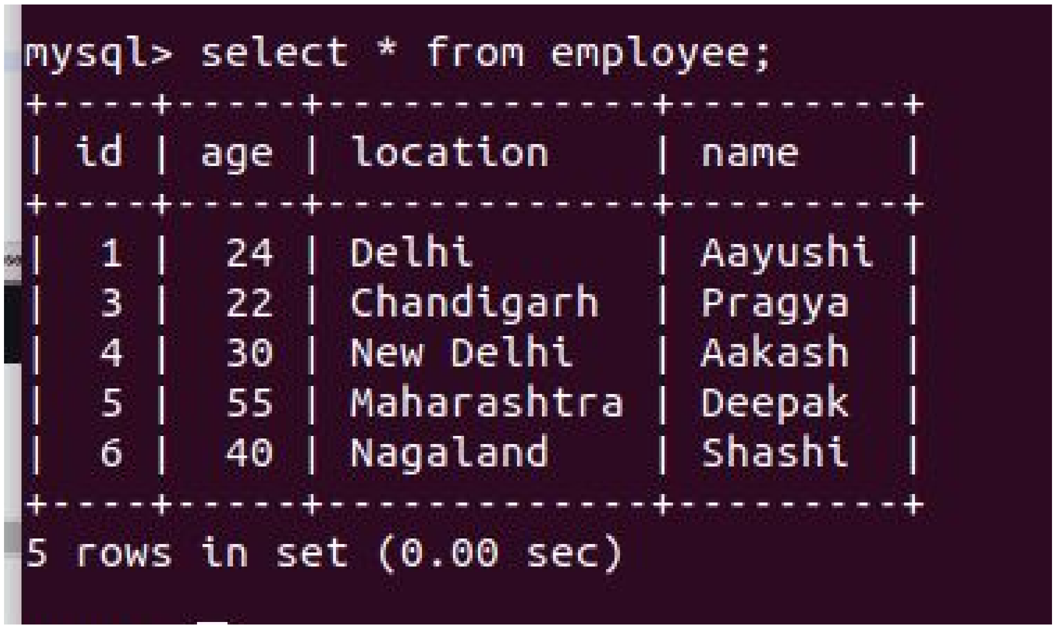
```
@Repository
public interface EmployeeRepository extends CrudRepository<Employee,Integer> {

   List<Employee> findByNameLike(String desc);
}
```

EmployeeController.java

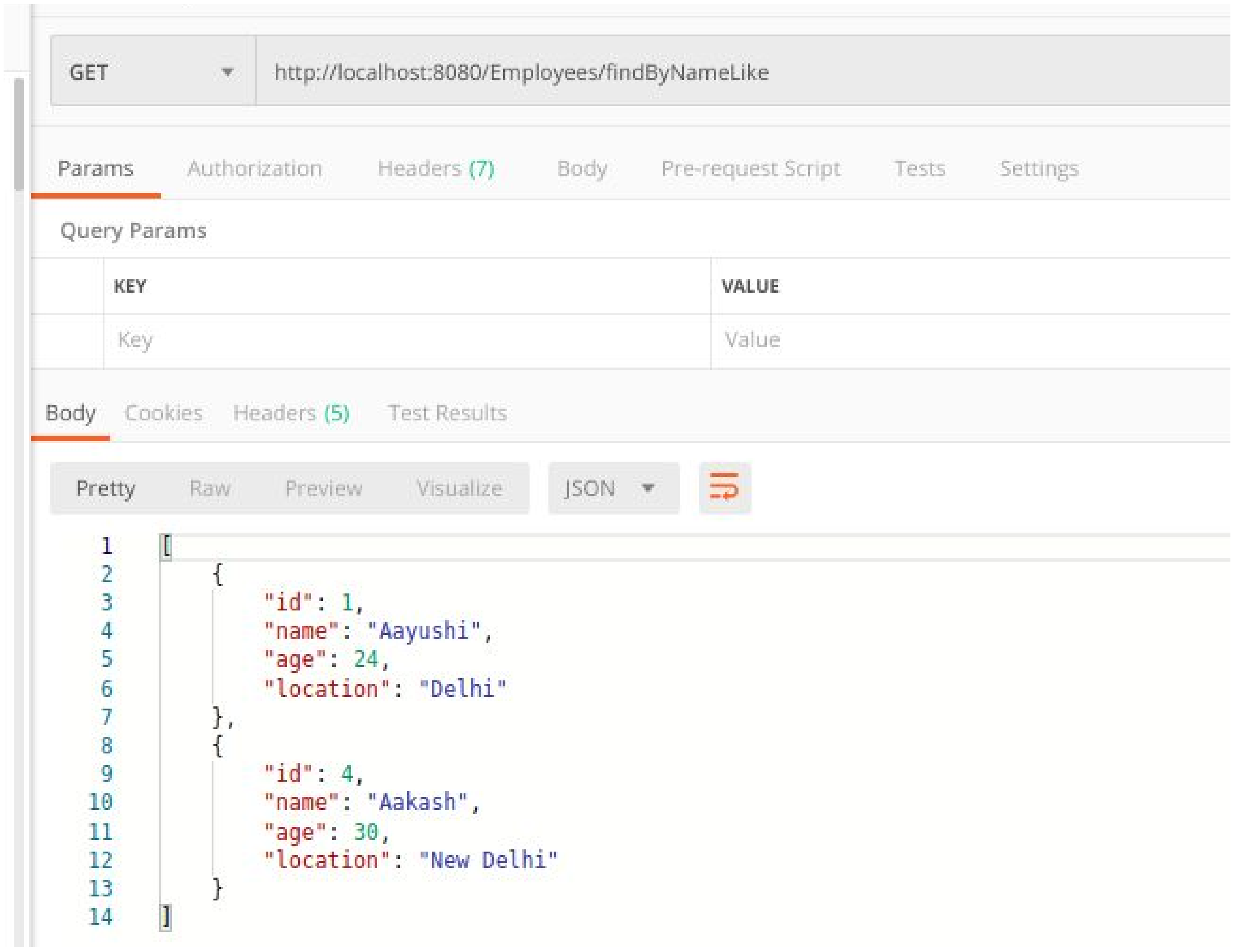
```
//FIND BY NAME STARTING WITH 'A' OPERATION
```

```
@GetMapping("/findByNameLike")
public List<Employee> findEmployeeByNameLike(){
   List<Employee> employeeList = (List<Employee>)
employeeRepository.findByNameLike("A%");
   return employeeList;
}
```

OUTPUT



```
mysql> select * from employee;
+----+-----+-------------+---------+
| id | age | location    | name    |
+----+-----+-------------+---------+
|  1 |  24 | Delhi       | Aayushi |
|  3 |  22 | Chandigarh  | Pragya  |
|  4 |  30 | New Delhi   | Aakash  |
|  5 |  55 | Maharashtra | Deepak  |
|  6 |  40 | Nagaland    | Shashi  |
+----+-----+-------------+---------+
5 rows in set (0.00 sec)
```



http://localhost:8080/Employees/findByNameLike

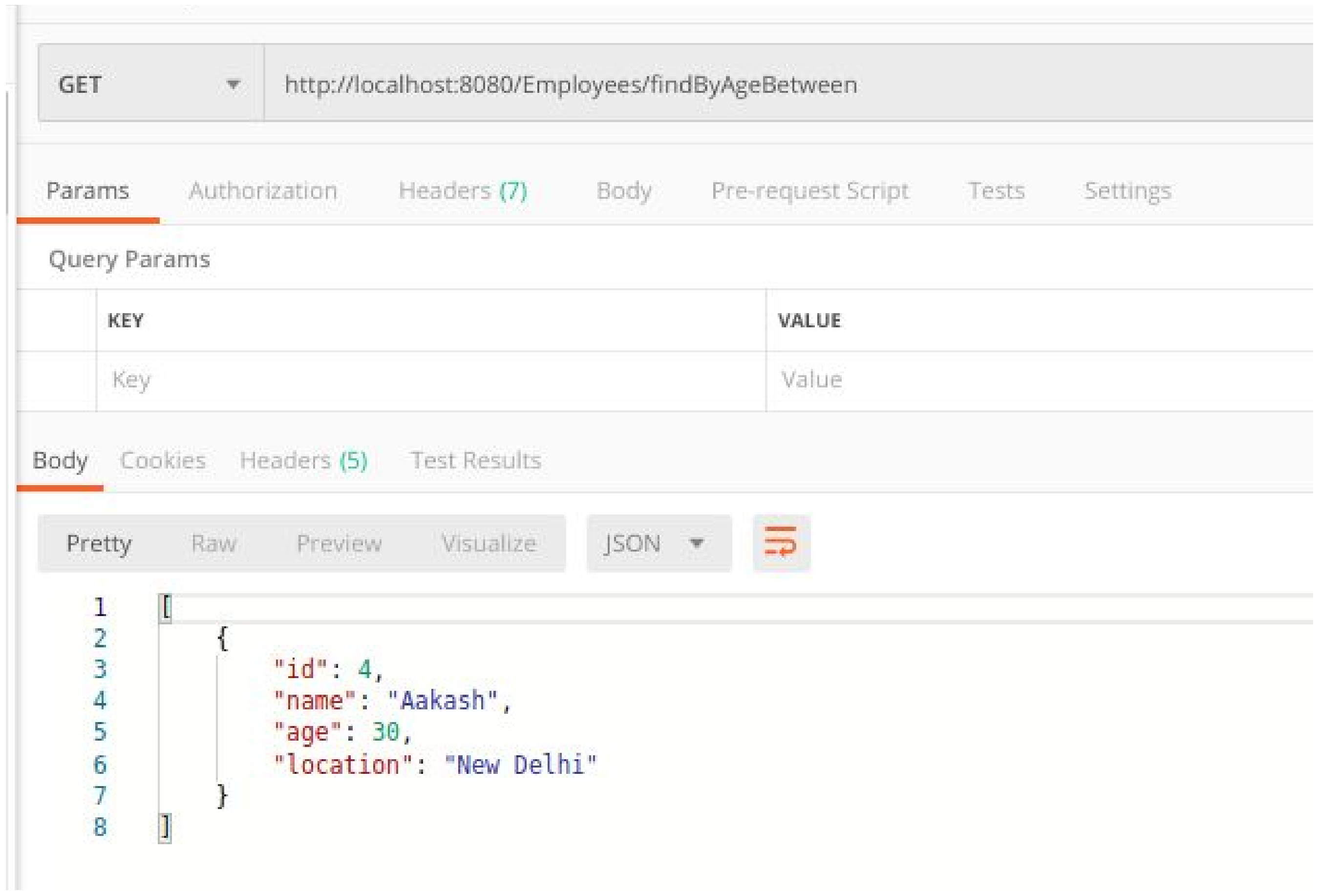
```
[
    {
        "id": 1,
        "name": "Aayushi",
        "age": 24,
        "location": "Delhi"
    },
    {
        "id": 4,
        "name": "Aakash",
        "age": 30,
        "location": "New Delhi"
    }
]
```

(10) Create and use finder to find Employees Between the age of 28 to 32.

CODE

```
//FIND BY AGE BETWEEN 28 TO 32
@GetMapping("/findByAgeBetween")
public List<Employee> findEmployeeByAgeBetween(){
  List<Employee> employeeList = (List<Employee>)
employeeRepository.findByAgeBetween(28,32);
  return employeeList;
}
```

OUTPUT



```
http://localhost:8080/Employees/findByAgeBetween
```

```
[
    {
        "id": 4,
        "name": "Aakash",
        "age": 30,
        "location": "New Delhi"
    }
]
```