SPRING BOOT

1) **Design a Spring Boot program to create a CRUD (Create, Read, Update, Delete) application using Hibernate for managing employee records. The program should allow users to perform the following operations on the employee database:**

   a) **Add a new employee: The user can enter details like employee name, department, and salary, and the program should add the employee to the database.**
   b) **Update employee details: The user can update the name, department, or salary of an existing employee based on their employee ID.**
   c) **Delete an employee: The user can delete an employee from the database based on their employee ID.**
   d) **Display all employees: The program should retrieve and display a list of all employees and their details from the database.**
   e) **Requirements:**
      i) **Use Spring Boot to create the application and Hibernate to manage the database.**
      ii) **Implement JPA (Java Persistence API) for data access.**
      iii) **Provide a RESTful API for performing CRUD operations on employees.**
      iv) **Implement exception handling to handle possible errors during database interactions.**
      v) **Cover Spring Boot and Hibernate topics, such as entity classes, repositories, services, and controllers.**
   f) **Note: Before running the program, make sure you have set up the database and configured the connection in the application.properties file.**

Code

Employee.java

```java
package in.springbootcrud.springbootcrudapi.model;

import java.sql.Date;

import jakarta.persistence.Entity;
import jakarta.persistence.GenerationType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Table;
import jakarta.persistence.Column;
import jakarta.persistence.Id;

@Entity
@Table(name = "TBL_employee")
public class Employee {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column
	private Integer id;
	@Column
	private String name ;
	@Column
	private String gender;
	@Column
	private String department;
	@Column
	private double salary;
	@Column
	private Date dob;
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getGender() {
		return gender;
	}
	public void setGender(String gender) {
		this.gender = gender;
	}
	public String getDepartment() {
		return department;
	}
	public void setDepartment(String department) {
		this.department = department;
	}
	public double getSalary() {
		return salary;
	}
	public void setSalary(double salary) {
		this.salary = salary;
	}
	public Date getDob() {
		return dob;
	}
	public void setDob(Date dob) {
		this.dob = dob;
	}
	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", gender=" + gender + ", department=" + department
				+ ", salary=" + salary + ", dob=" + dob + ", getId()=" + getId() + ", getName()=" + getName()
				+ ", getGender()=" + getGender() + ", getDepartment()=" + getDepartment() + ", getSalary()="
				+ getSalary() + ", getDob()=" + getDob() + ", getClass()=" + getClass() + ", hashCode()=" + hashCode()
				+ ", toString()=" + super.toString() + "]";
	}

}
```

EmployeeDAO.java

```
package in.springbootcrud.springbootcrudapi.deo;

import java.util.List;

import in.springbootcrud.springbootcrudapi.model.*;

public interface EmployeeDAO {

	List<Employee> get();

	Employee get(int id);
	void save(Employee employee);

	void delete(int id);
}
```

EmployeeDAOImpl.java

```
package in.springbootcrud.springbootcrudapi.deo;

import java.util.List;

import jakarta.persistence.EntityManager;

import org.hibernate.Session;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import in.springbootcrud.springbootcrudapi.model.Employee;

@Repository
public class EmploeeDAOImpl implements EmployeeDAO {

	@Autowired
	private EntityManager entityManager;

	@Override
	public List<Employee> get() {
		Session currentSession = entityManager.unwrap(Session.class);
		Query<Employee> query = currentSession.createQuery("from Employee",
Employee.class);
		List<Employee> list = query.getResultList();
		return list;
	}

	@Override
	public Employee get(int id) {
		Session currentSession = entityManager.unwrap(Session.class);
		Employee employeeObj = currentSession.get(Employee.class, id);
		return employeeObj;
	}

	@Override
	public void save(Employee employee) {
		Session currentSession = entityManager.unwrap(Session.class);
		currentSession.saveOrUpdate(employee);

	}

	@Override
	public void delete(int id) {
		Session currentSession = entityManager.unwrap(Session.class);
		Employee employeeObj = currentSession.get(Employee.class, id);
		currentSession.delete(employeeObj);
	}

}
```

EmployeeService.java

```
package in.springbootcrud.springbootcrudapi.service;

import java.util.List;

import in.springbootcrud.springbootcrudapi.model.Employee;

public interface EmployeeService {

	List<Employee> get();

	Employee get(int id);

	void save(Employee employee);

	void delete(int id);
}
```

EmployeeServiceImpl.java

```
package in.springbootcrud.springbootcrudapi.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```java
import org.springframework.transaction.annotation.Transactional;

import in.springbootcrud.springbootcrudapi.deo.EmployeeDAO;
import in.springbootcrud.springbootcrudapi.model.Employee;

@Service
public class EmployeeServiceImpl implements EmployeeService {

	@Autowired
	private EmployeeDAO employeeDAO;

	@Transactional
	@Override
	public List<Employee> get() {
		return employeeDAO.get();
	}

	@Transactional
	@Override
	public Employee get(int id) {
		return employeeDAO.get(id);
	}

	@Transactional
	@Override
	public void save(Employee employee) {
		employeeDAO.save(employee);

	}

	@Transactional
	@Override
	public void delete(int id) {
		employeeDAO.delete(id);

	}

}
```

EmployeeController.java

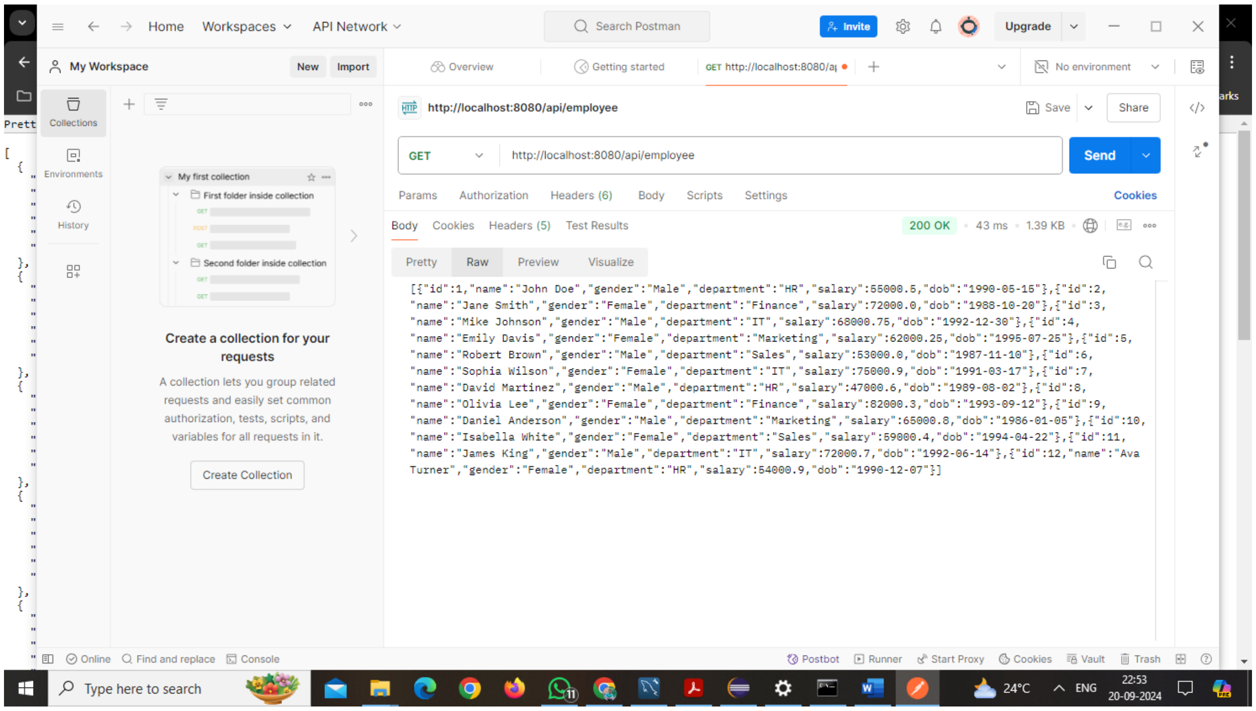
```java
package in.springbootcrud.springbootcrudapi.controller;
```

```
import java.util.List;
import java.io.IOException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import in.springbootcrud.springbootcrudapi.model.Employee;
import in.springbootcrud.springbootcrudapi.service.EmployeeService;

@RestController
@RequestMapping("/api")
public class EmployeeController {

	@Autowired
	private EmployeeService employeeService;

	@GetMapping("/employee")
	public List<Employee> get(){
		return employeeService.get();
	}

	@PostMapping("/employee")
	public Employee save(@RequestBody Employee employeeObj) {
		employeeService.save(employeeObj);
		return employeeObj;
	}

	@GetMapping("/employee/{id}")
	public Employee get(@PathVariable int id) {
		Employee employeeObj = employeeService.get(id);
		if(employeeObj == null) {
			throw new RuntimeException("Employee with id "+id+" is not found");
		}
		return employeeObj;

	}
```

```
        @DeleteMapping("/employee/{id}")
        public String delete(@PathVariable int id) {
                employeeService.delete(id);
                return "Emloyee has been deleted with id:"+id;
        }

        @PutMapping("/employee")
        public Employee update(@RequestBody Employee employeeObj) {
                employeeService.save(employeeObj);
                return employeeObj;
        }
}
```

- Output :

- **Display all employees: The program should retrieve and display a list of all employees and their details from the database.**
  By using **Postman** to send a **GET** request to the API endpoint http://localhost:8080/api/employee to retrieve and display all employee records from the database.



http://localhost:8080/api/employee

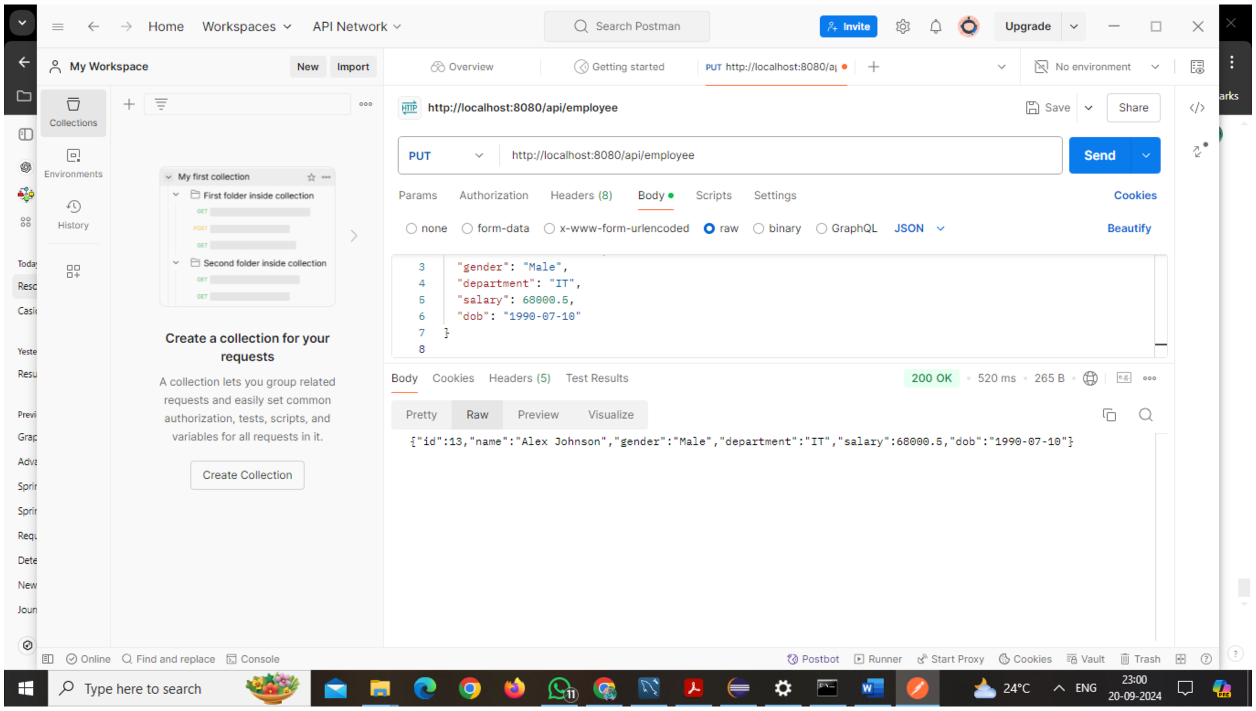
GET http://localhost:8080/api/employee

200 OK 43 ms 1.39 KB

```
[{"id":1,"name":"John Doe","gender":"Male","department":"HR","salary":55000.5,"dob":"1990-05-15"},{"id":2,
"name":"Jane Smith","gender":"Female","department":"Finance","salary":72000.0,"dob":"1988-10-20"},{"id":3,
"name":"Mike Johnson","gender":"Male","department":"IT","salary":68000.75,"dob":"1992-12-30"},{"id":4,
"name":"Emily Davis","gender":"Female","department":"Marketing","salary":62000.25,"dob":"1995-07-25"},{"id":5,
"name":"Robert Brown","gender":"Male","department":"Sales","salary":53000.0,"dob":"1987-11-10"},{"id":6,
"name":"Sophia Wilson","gender":"Female","department":"IT","salary":75000.9,"dob":"1991-03-17"},{"id":7,
"name":"David Martinez","gender":"Male","department":"HR","salary":47000.6,"dob":"1989-08-02"},{"id":8,
"name":"Olivia Lee","gender":"Female","department":"Finance","salary":82000.3,"dob":"1993-09-12"},{"id":9,
"name":"Daniel Anderson","gender":"Male","department":"Marketing","salary":65000.8,"dob":"1986-01-05"},{"id":10,
"name":"Isabella White","gender":"Female","department":"Sales","salary":59000.4,"dob":"1994-04-22"},{"id":11,
"name":"James King","gender":"Male","department":"IT","salary":72000.7,"dob":"1992-06-14"},{"id":12,"name":"Ava
Turner","gender":"Female","department":"HR","salary":54000.9,"dob":"1990-12-07"}]
```

➢ Add a new employee: The user can enter details like employee name, department, and salary, and the program should add the employee to the database.

After adding the new employee using the POST method, the API automatically assigns an ID to the new entry. In this case, the employee Alex Johnson has been assigned the ID 13



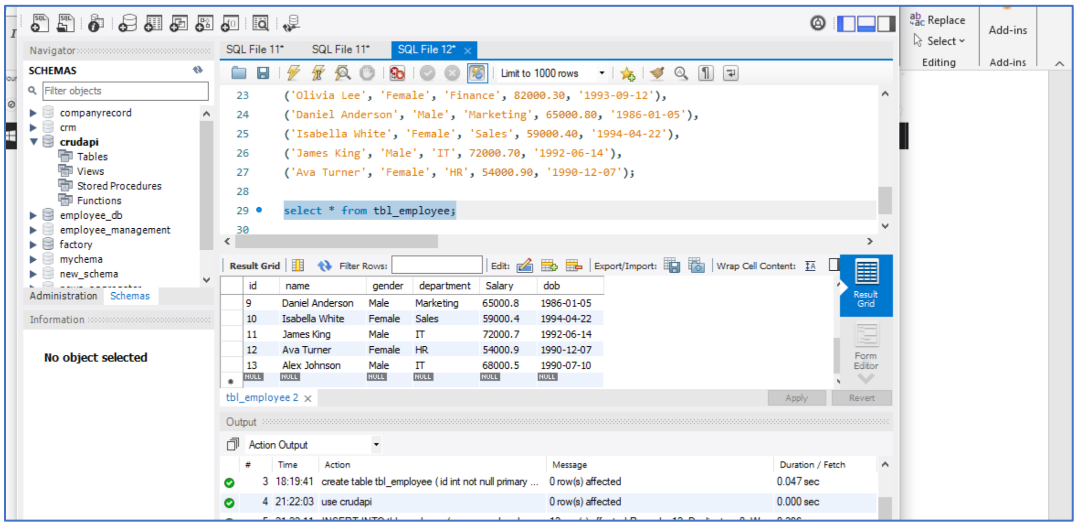
PUT http://localhost:8080/api/employee

```
  "gender": "Male",
  "department": "IT",
  "salary": 68000.5,
  "dob": "1990-07-10"
}
```

200 OK · 520 ms · 265 B

```
{"id":13,"name":"Alex Johnson","gender":"Male","department":"IT","salary":68000.5,"dob":"1990-07-10"}
```

The new entry will added in emp database id -- 13



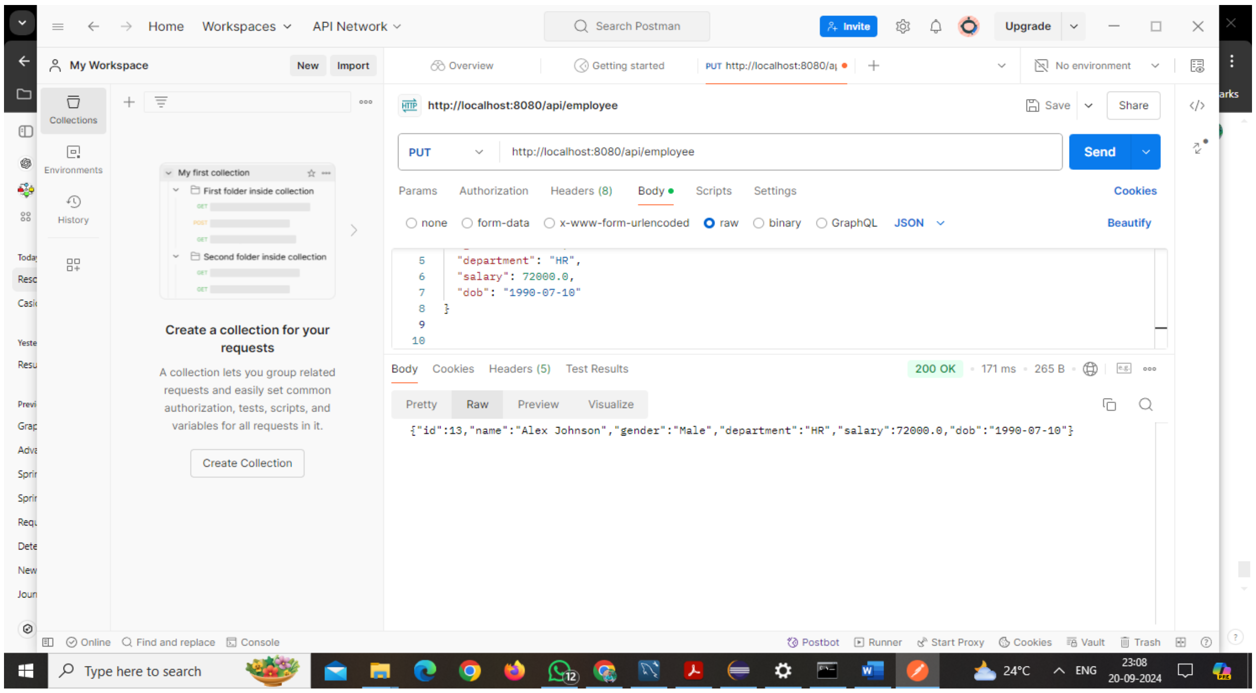
```
    ('Olivia Lee', 'Female', 'Finance', 82000.30, '1993-09-12'),
    ('Daniel Anderson', 'Male', 'Marketing', 65000.80, '1986-01-05'),
    ('Isabella White', 'Female', 'Sales', 59000.40, '1994-04-22'),
    ('James King', 'Male', 'IT', 72000.70, '1992-06-14'),
    ('Ava Turner', 'Female', 'HR', 54000.90, '1990-12-07');

select * from tbl_employee;
```

id	name	gender	department	Salary	dob
9	Daniel Anderson	Male	Marketing	65000.8	1986-01-05
10	Isabella White	Female	Sales	59000.4	1994-04-22
11	James King	Male	IT	72000.7	1992-06-14
12	Ava Turner	Female	HR	54000.9	1990-12-07
13	Alex Johnson	Male	IT	68000.5	1990-07-10
NULL	NULL	NULL	NULL	NULL	NULL

Action Output

#	Time	Action	Message	Duration / Fetch
3	18:19:41	create table tbl_employee (id int not null primary ...	0 row(s) affected	0.047 sec
4	21:22:03	use crudapi	0 row(s) affected	0.000 sec

- Update employee details: The user can update the name, department, or salary of an existing employee based on their employee ID.
  In that we update emp id 13 department from ‘IT’ to ‘HR’ and Increment a salary from ‘68000.00’ to ‘72000.00’



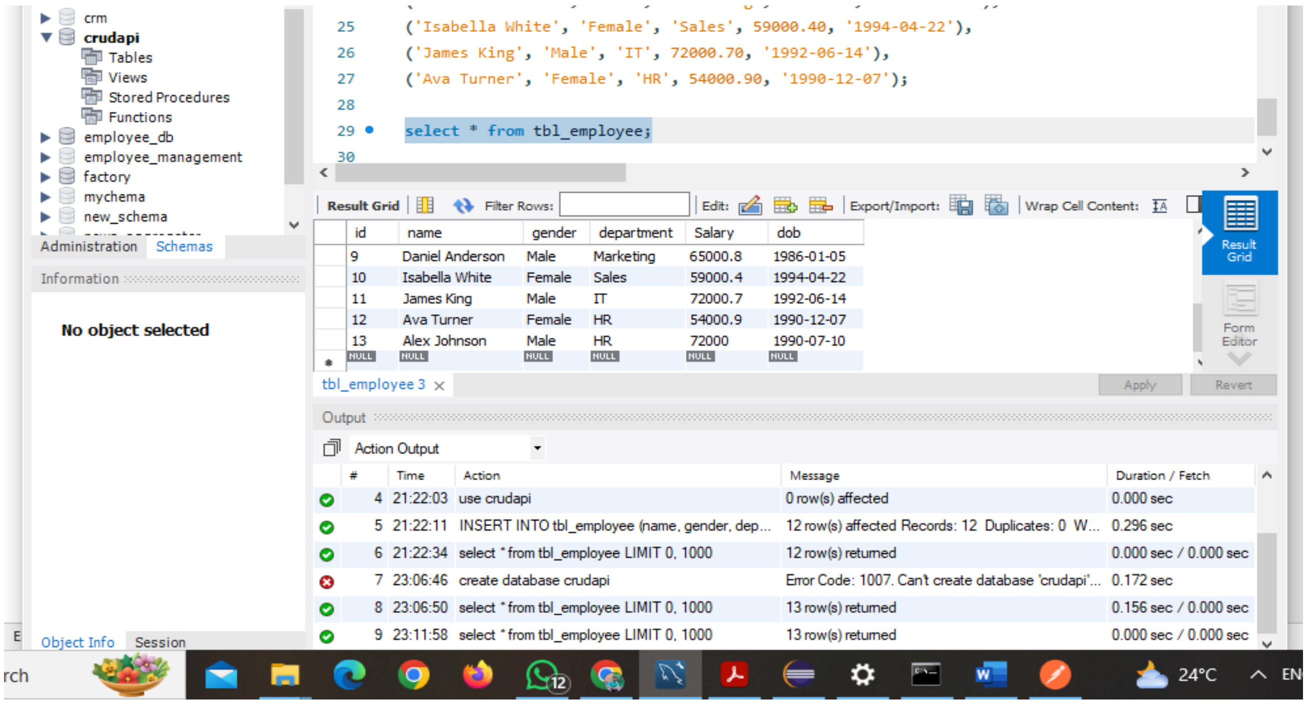
PUT http://localhost:8080/api/employee

Body (raw, JSON):

```
  "department": "HR",
  "salary": 72000.0,
  "dob": "1990-07-10"
}
```

Response: 200 OK · 171 ms · 265 B

```
{"id":13,"name":"Alex Johnson","gender":"Male","department":"HR","salary":72000.0,"dob":"1990-07-10"}
```



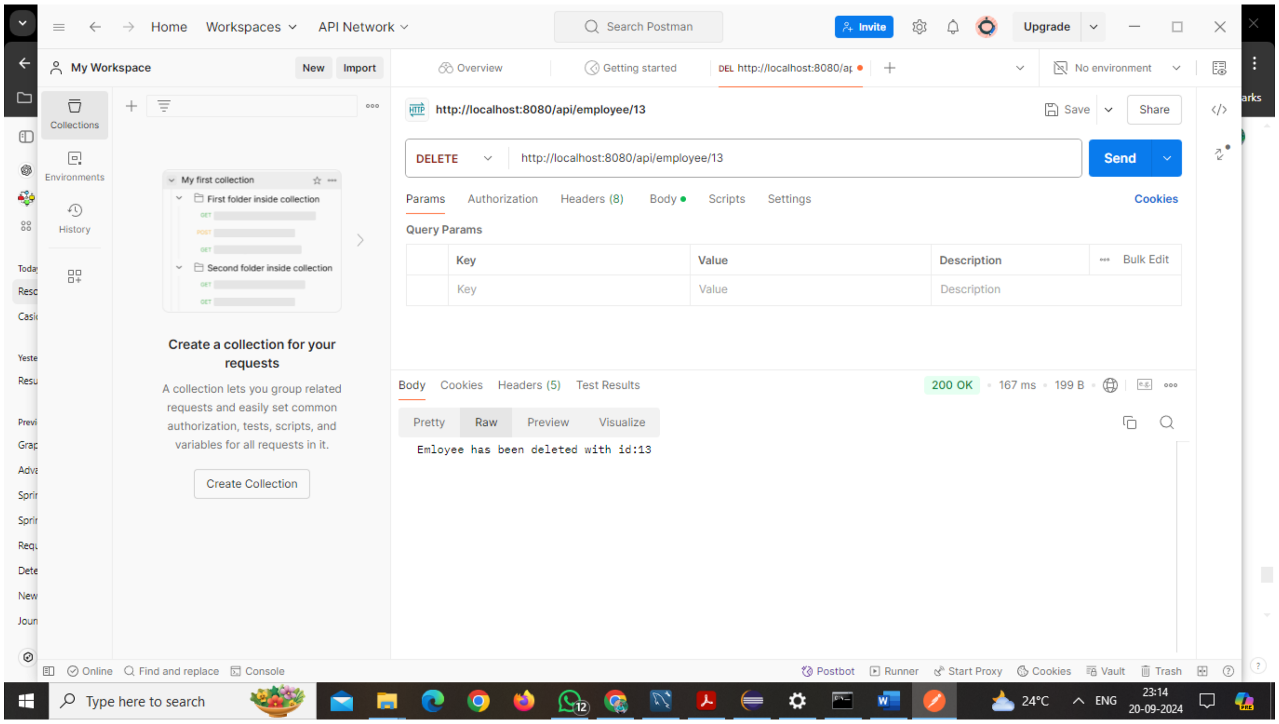
```
('Isabella White', 'Female', 'Sales', 59000.40, '1994-04-22'),
('James King', 'Male', 'IT', 72000.70, '1992-06-14'),
('Ava Turner', 'Female', 'HR', 54000.90, '1990-12-07');

select * from tbl_employee;
```

id	name	gender	department	Salary	dob
9	Daniel Anderson	Male	Marketing	65000.8	1986-01-05
10	Isabella White	Female	Sales	59000.4	1994-04-22
11	James King	Male	IT	72000.7	1992-06-14
12	Ava Turner	Female	HR	54000.9	1990-12-07
13	Alex Johnson	Male	HR	72000	1990-07-10
NULL	NULL	NULL	NULL	NULL	NULL

Action Output

#	Time	Action	Message	Duration / Fetch
4	21:22:03	use crudapi	0 row(s) affected	0.000 sec
5	21:22:11	INSERT INTO tbl_employee (name, gender, dep...	12 row(s) affected Records: 12 Duplicates: 0 W...	0.296 sec
6	21:22:34	select * from tbl_employee LIMIT 0, 1000	12 row(s) returned	0.000 sec / 0.000 sec
7	23:06:46	create database crudapi	Error Code: 1007. Can't create database 'crudapi'...	0.172 sec
8	23:06:50	select * from tbl_employee LIMIT 0, 1000	13 row(s) returned	0.156 sec / 0.000 sec
9	23:11:58	select * from tbl_employee LIMIT 0, 1000	13 row(s) returned	0.000 sec / 0.000 sec

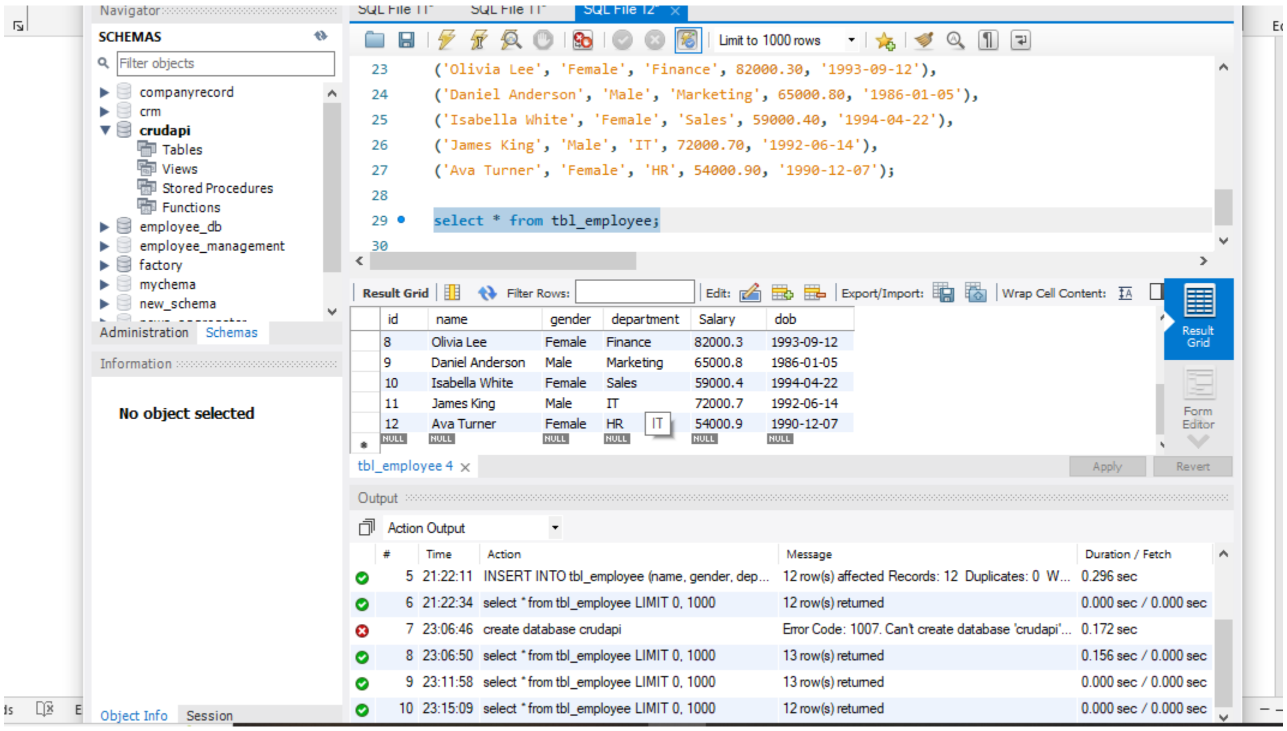
➢ Delete an employee: The user can delete an employee from the database based on their employee ID.
In that we delete emp id 13 records.



DELETE http://localhost:8080/api/employee/13

200 OK 167 ms 199 B

```
Emloyee has been deleted with id:13
```



```
('Olivia Lee', 'Female', 'Finance', 82000.30, '1993-09-12'),
('Daniel Anderson', 'Male', 'Marketing', 65000.80, '1986-01-05'),
('Isabella White', 'Female', 'Sales', 59000.40, '1994-04-22'),
('James King', 'Male', 'IT', 72000.70, '1992-06-14'),
('Ava Turner', 'Female', 'HR', 54000.90, '1990-12-07');

select * from tbl_employee;
```

id	name	gender	department	Salary	dob
8	Olivia Lee	Female	Finance	82000.3	1993-09-12
9	Daniel Anderson	Male	Marketing	65000.8	1986-01-05
10	Isabella White	Female	Sales	59000.4	1994-04-22
11	James King	Male	IT	72000.7	1992-06-14
12	Ava Turner	Female	HR	54000.9	1990-12-07
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message	Duration / Fetch
5	21:22:11	INSERT INTO tbl_employee (name, gender, dep...	12 row(s) affected Records: 12 Duplicates: 0 W...	0.296 sec
6	21:22:34	select * from tbl_employee LIMIT 0, 1000	12 row(s) returned	0.000 sec / 0.000 sec
7	23:06:46	create database crudapi	Error Code: 1007. Can't create database 'crudapi'...	0.172 sec
8	23:06:50	select * from tbl_employee LIMIT 0, 1000	13 row(s) returned	0.156 sec / 0.000 sec
9	23:11:58	select * from tbl_employee LIMIT 0, 1000	13 row(s) returned	0.000 sec / 0.000 sec
10	23:15:09	select * from tbl_employee LIMIT 0, 1000	12 row(s) returned	0.000 sec / 0.000 sec