**WEEK-3**

**SPRING DATA JPA AND HIBERNATE**

**Exercise 1: Employee Management System - Overview and Setup:**

**CODE:**

**EmployeeManagementSystemApplication.java:**

EmployeeManagementSystemApplication.java:  
package com.example.employeemanagementsystem;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

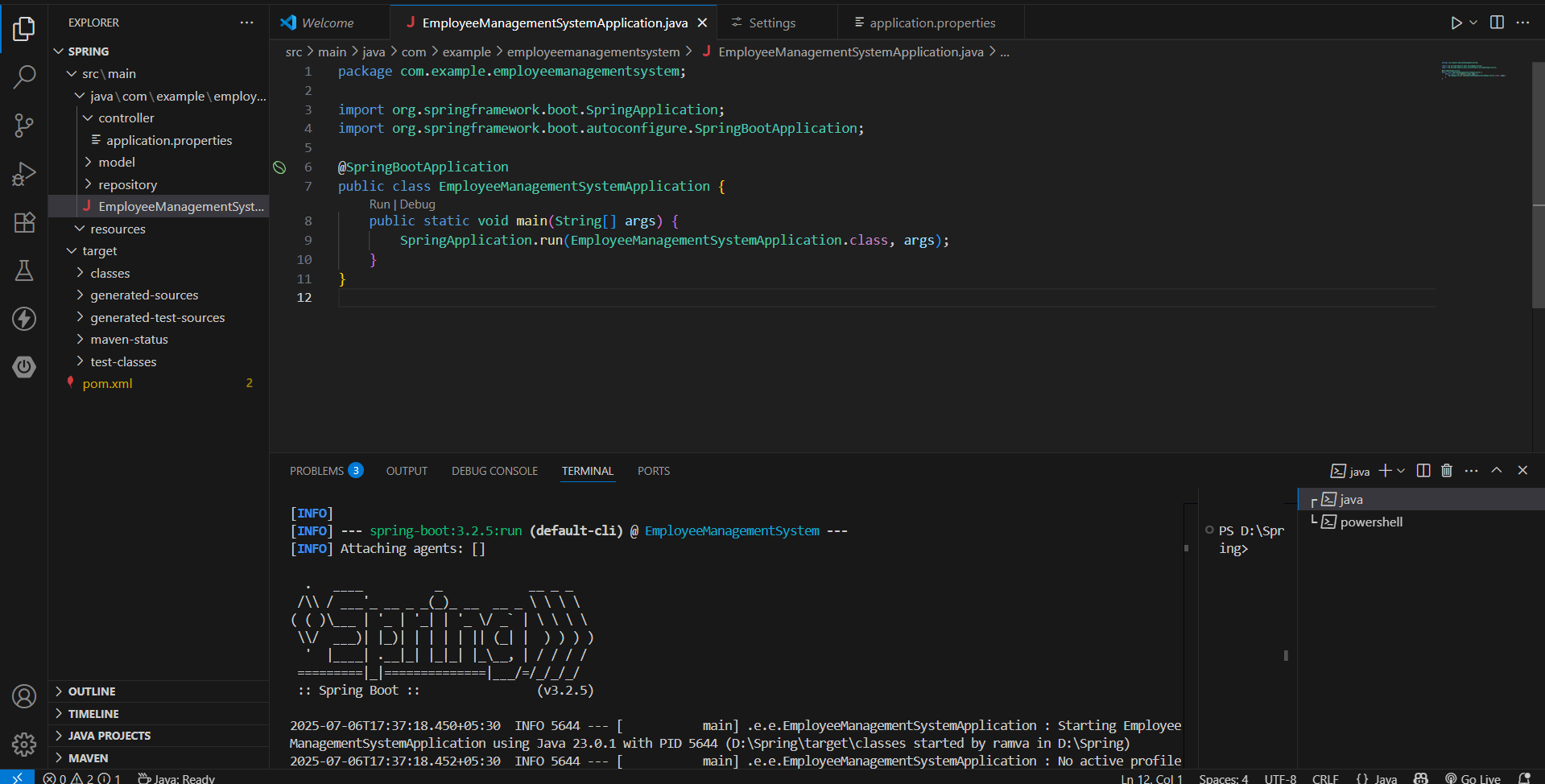
    public static void main(String[] args) {

        SpringApplication.run(EmployeeManagementSystemApplication.class, args);

    }

}

**OUTPUT:**



**Exercise 2: Employee Management System - Creating Entities**

**CODE:**

**HelloController.java:**

package com.example.employeemanagementsystem.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

    @GetMapping("/")

    public String home() {

        return " Employee Management System is running!";

    }

}

**EmployeeManagementSystemApplication.java:**

package com.example.employeemanagementsystem;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

    public static void main(String[] args) {

        SpringApplication.run(EmployeeManagementSystemApplication.class, args);

    }

}

Department.java:  
package com.example.employeemanagementsystem.model;

import jakarta.persistence.\*;

import lombok.\*;

import java.util.List;

@Entity

@Table(name = "departments")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Department

@Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    @OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

    private List<Employee> employees;

}

**Employee.java:**

package com.example.employeemanagementsystem.model;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Table(name = "employees")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private String email;

    // Many employees belong to one department

    @ManyToOne

    @JoinColumn(name = "department\_id")

    private Department department;

}

**OUTPUT:**



**Exercise 3: Employee Management System - Creating Repositories**

**CODE:**

**DepartmentRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Department;

import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Long> {

    Department findByName(String name);

}

**EmployeeRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

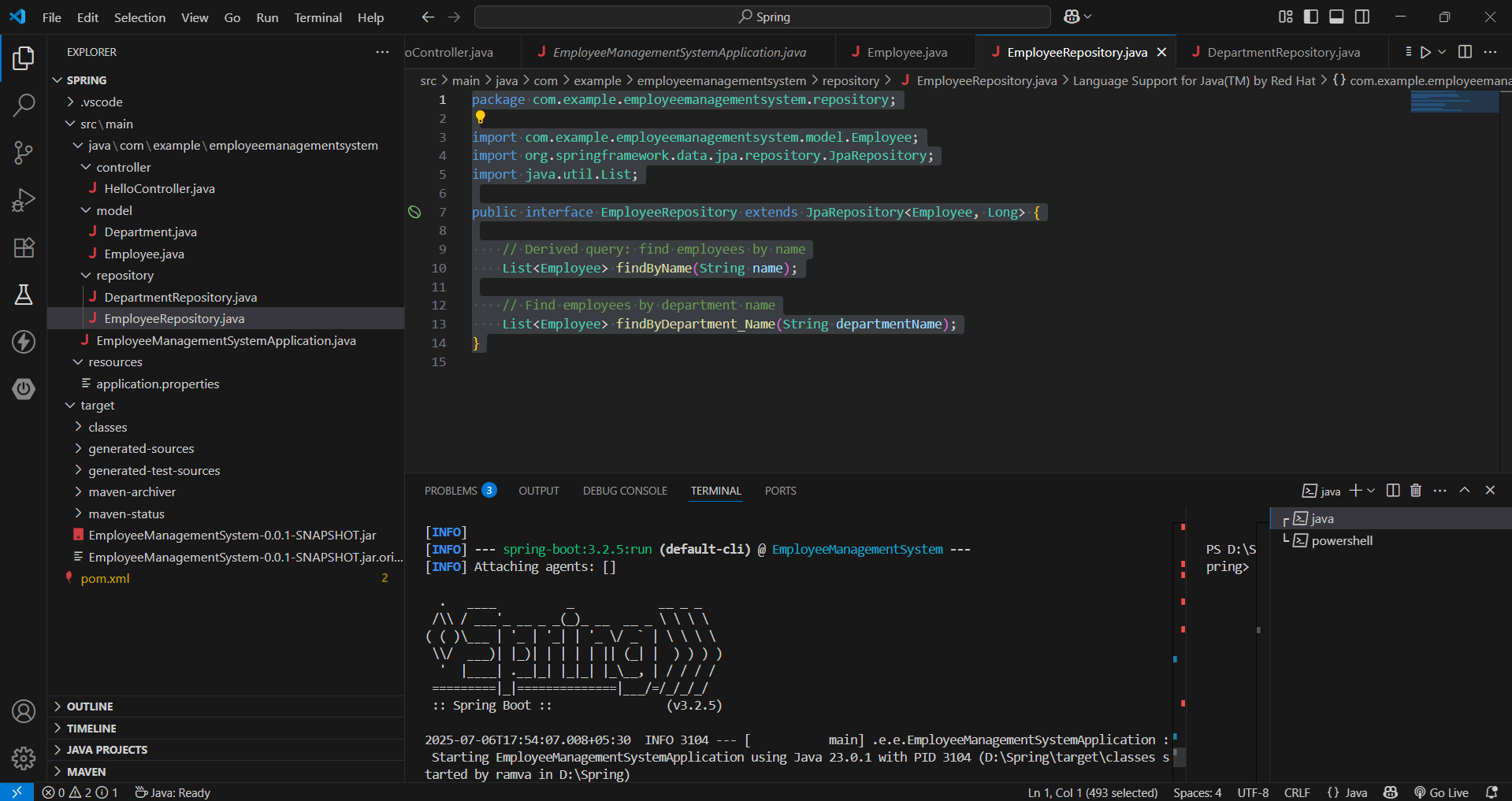
public interface EmployeeRepository extends JpaRepository<Employee, Long> {

    List<Employee> findByName(String name);

    List<Employee> findByDepartment\_Name(String departmentName);

}

**OUTPUT:**

****

**Exercise 4: Employee Management System - Implementing CRUD Operations**

**CODE:**

**DepartmentController.java:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Department;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/departments")

public class DeptController {

@Autowired

private DepartmentRepository deptRepo;

@GetMapping

public List<Department> getAllDepartments() {

return deptRepo.findAll();

}

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable Long id) {

return deptRepo.findById(id).orElse(null);

}

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return deptRepo.save(department);

}

@PutMapping("/{id}")

public Department updateDepartment(@PathVariable Long id, @RequestBody Department departmentDetails) {

Department department = deptRepo.findById(id).orElse(null);

if (department != null) {

department.setName(departmentDetails.getName());

return deptRepo.save(department);

}

return null;

}

@DeleteMapping("/{id}")

public void deleteDepartment(@PathVariable Long id) {

deptRepo.deleteById(id);

}

}

**EmployeeController.java:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmpController {

@Autowired

private EmployeeRepository employeeRepo;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepo.findAll();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable Long id) {

return employeeRepo.findById(id).orElse(null);

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepo.save(employee);

}

@PutMapping("/{id}")

public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails) {

Employee employee = employeeRepo.findById(id).orElse(null);

if (employee != null) {

employee.setName(employeeDetails.getName());

employee.setEmail(employeeDetails.getEmail());

employee.setDepartment(employeeDetails.getDepartment());

return employeeRepo.save(employee);

}

return null;

}

@DeleteMapping("/{id}")

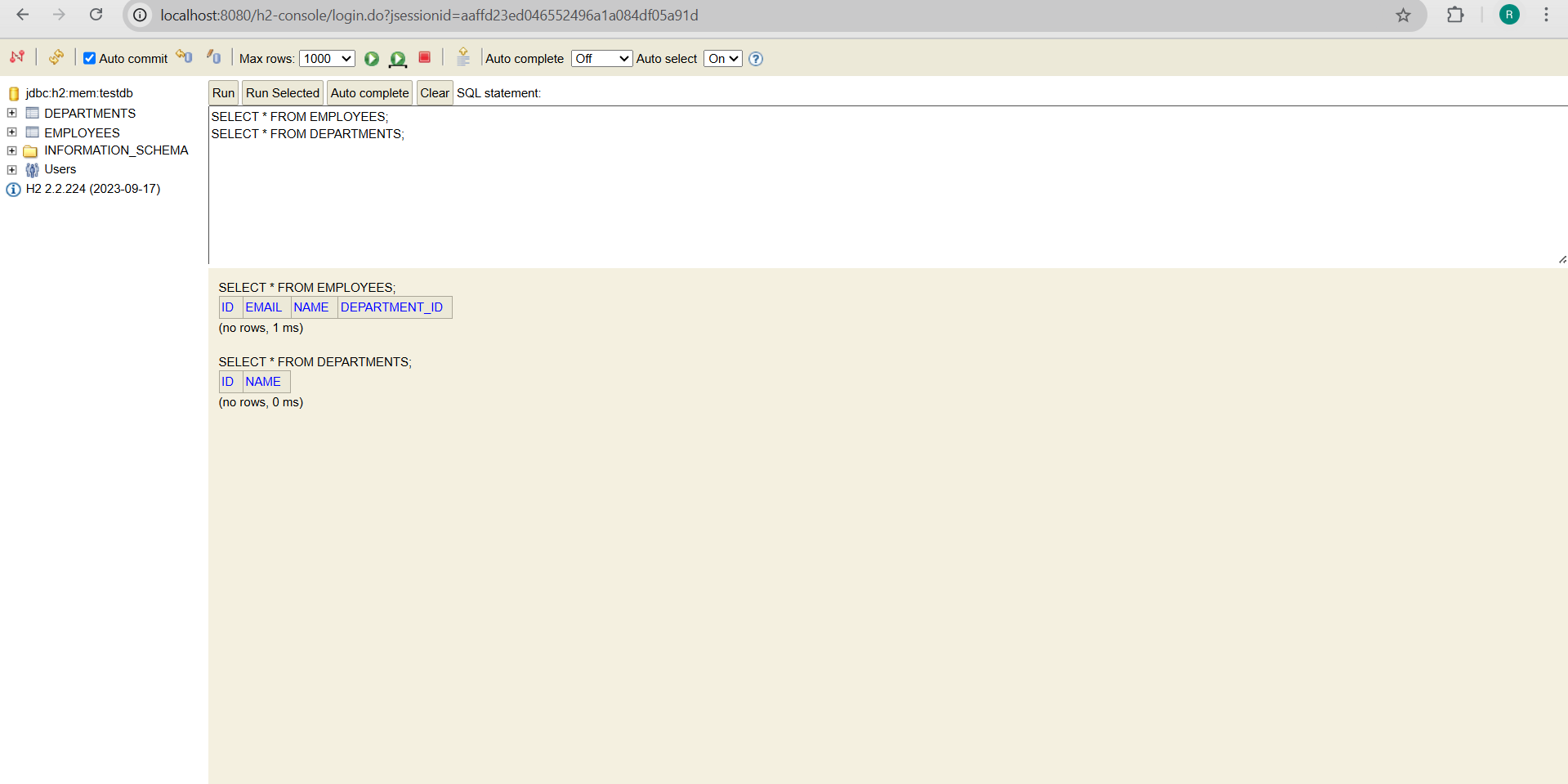
public void deleteEmployee(@PathVariable Long id) {

employeeRepo.deleteById(id);

}

}

**OUTPUT:**



**Exercise 5: Employee Management System - Defining Query Methods.**

**CODE:**

**DepartmentRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Department;

import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Long> {

    Department findByName(String name);

}

**EmployeeRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

    List<Employee> findByName(String name);

    List<Employee> findByDepartment\_Name(String departmentName);

    @Query("SELECT e FROM Employee e WHERE e.email = :email")

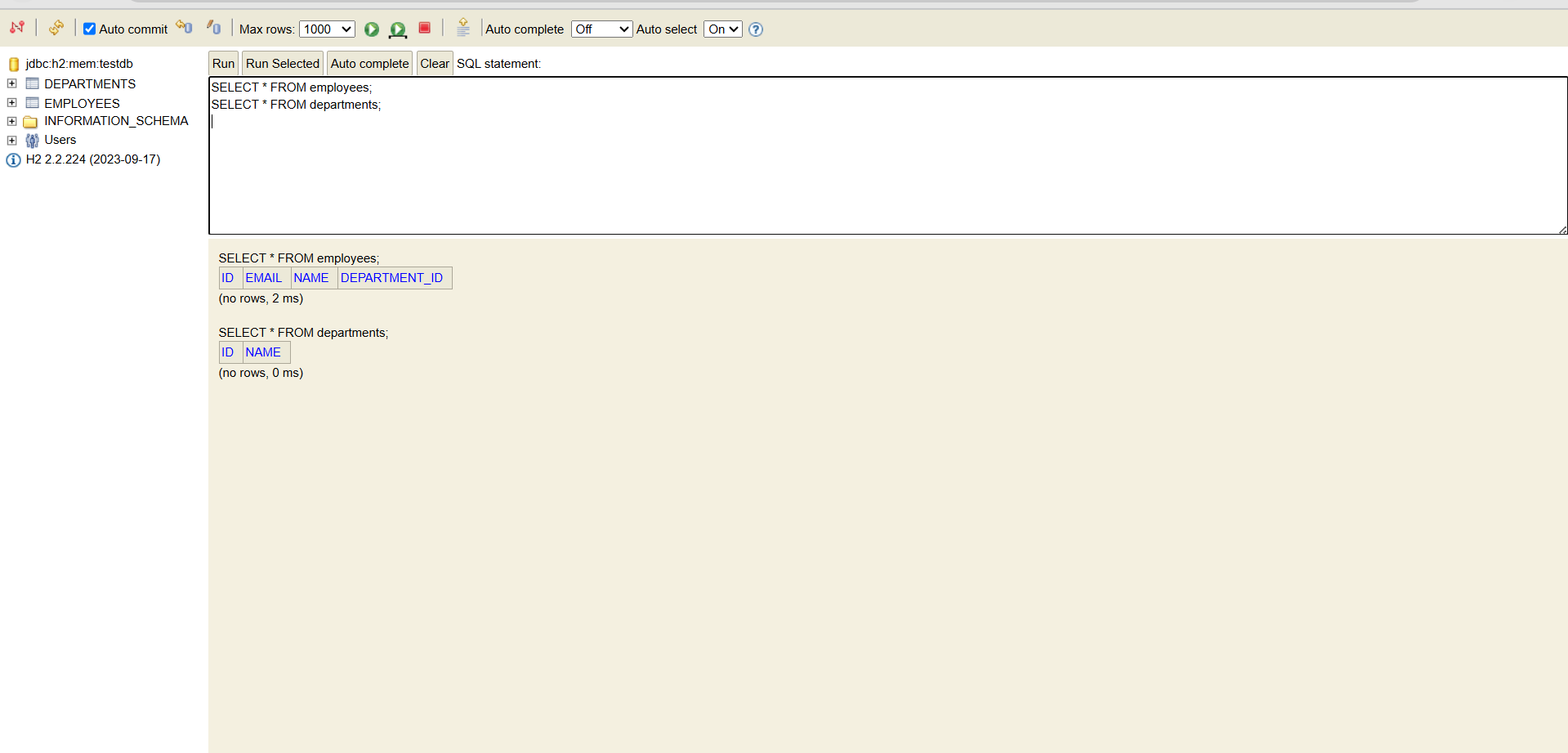
    List<Employee> findByEmail(@Param("email") String email);

    @Query("SELECT e.email FROM Employee e")

    List<String> getAllEmployeeEmails();

}

**OUTPUT:**

****

**Exercise 6: Employee Management System - Implementing Pagination and Sorting**

**CODE:**

**EmployeeController.java:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.\*;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmpController {

    @Autowired

    private EmployeeRepository employeeRepo;

    @GetMapping

    public List<Employee> getAllEmployees() {

        return employeeRepo.findAll();

    }

    @GetMapping("/{id}")

    public Employee getEmployeeById(@PathVariable Long id) {

        return employeeRepo.findById(id).orElse(null);

    }

    @PostMapping

    public Employee createEmployee(@RequestBody Employee employee) {

        return employeeRepo.save(employee);

    }

    @PutMapping("/{id}")

    public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails) {

        Employee employee = employeeRepo.findById(id).orElse(null);

        if (employee != null) {

            employee.setName(employeeDetails.getName());

            employee.setEmail(employeeDetails.getEmail());

            employee.setDepartment(employeeDetails.getDepartment());

            return employeeRepo.save(employee);

        }

        return null;

    }

    @DeleteMapping("/{id}")

    public void deleteEmployee(@PathVariable Long id) {

        employeeRepo.deleteById(id);

    }

    @GetMapping("/paginated")

    public Page<Employee> getPaginatedEmployees(

            @RequestParam(defaultValue = "0") int page,

            @RequestParam(defaultValue = "5") int size,

            @RequestParam(defaultValue = "id,asc") String[] sort) {

        Sort.Direction sortDirection = Sort.Direction.fromString(sort[1]);

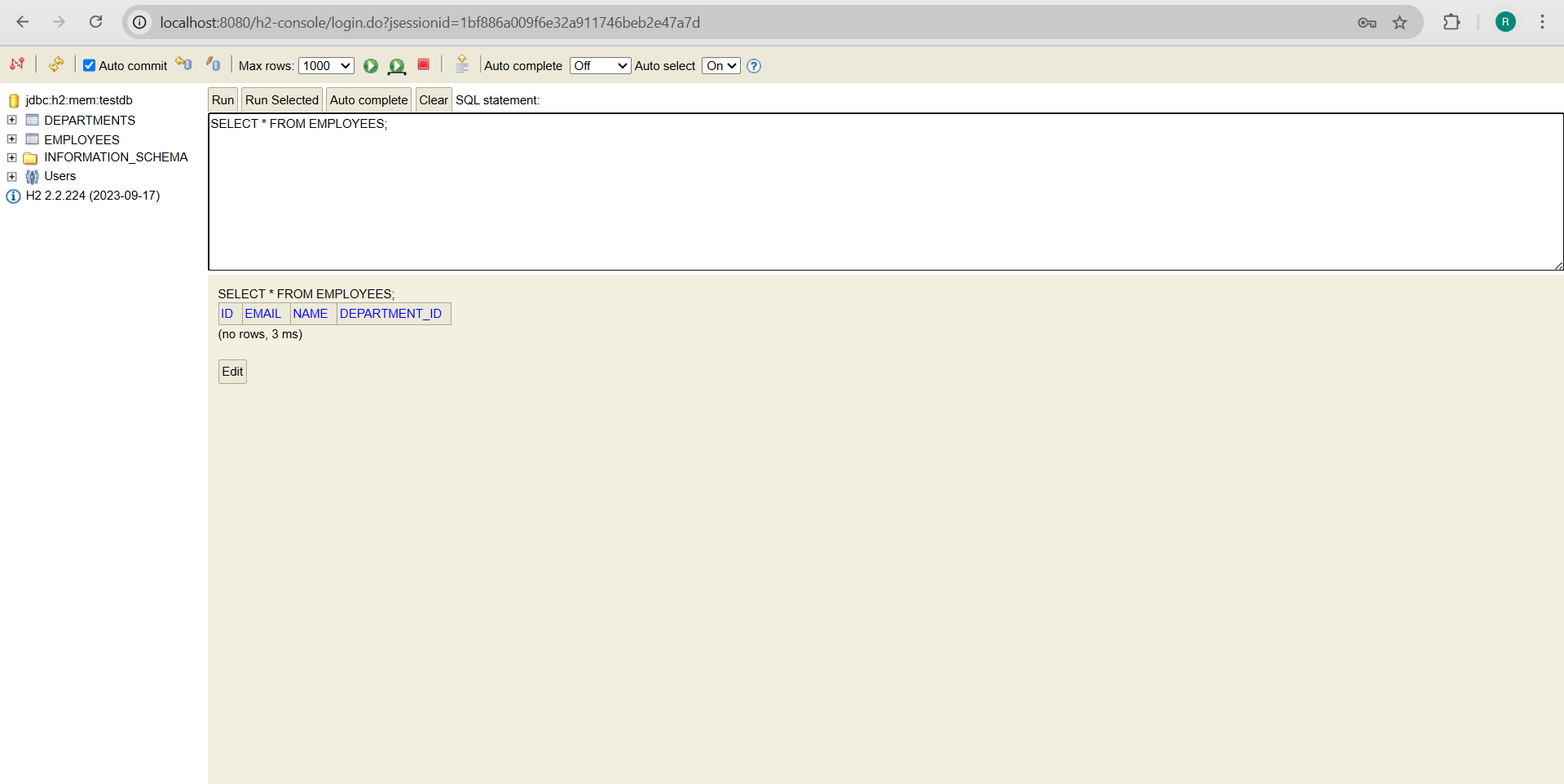
        Pageable pageable = PageRequest.of(page, size, Sort.by(sortDirection, sort[0]));

        return employeeRepo.findAll(pageable);

    }

}

**OUTPUT:**

****

**Exercise 7: Employee Management System - Enabling Entity Auditing.**

**CODE:**

EmployeeManagementSystemApplication.java:  
package com.example.employeemanagementsystem.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@Configuration

@EnableJpaAuditing

public class AuditingConfig {

}

AuditingConfig.java:  
package com.example.employeemanagementsystem.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@Configuration

@EnableJpaAuditing

public class AuditingConfig {

}

**OUTPUT:**

