**WEEK-3**

**SPRING DATA JPA AND HIBERNATE**

**Superset ID: 6419740**

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

**Code:**

**EmployeeRepository.java:**

package com.company.repository;

import com.company.entity.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**EmployeeController.java:**

package com.company.controller;

import com.company.entity.Employee;

import com.company.repository.EmployeeRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@PostMapping

public Employee create(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@GetMapping

public List<Employee> getAll() {

return employeeRepository.findAll();

}

@GetMapping("/{id}")

public Employee getById(@PathVariable Long id) {

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

}

@PutMapping("/{id}")

public Employee update(@PathVariable Long id, @RequestBody Employee updated) {

Employee emp = employeeRepository.findById(id).orElse(null);

if (emp != null) {

emp.setName(updated.getName());

emp.setDepartment(updated.getDepartment());

emp.setEmail(updated.getEmail());

return employeeRepository.save(emp);

}

return null;

}

@DeleteMapping("/{id}")

public void delete(@PathVariable Long id) {

employeeRepository.deleteById(id);

}

}

**Employee.java**

 package com.company.entity;

import jakarta.persistence.\*;

import lombok.Data;

@Entity

@Data

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

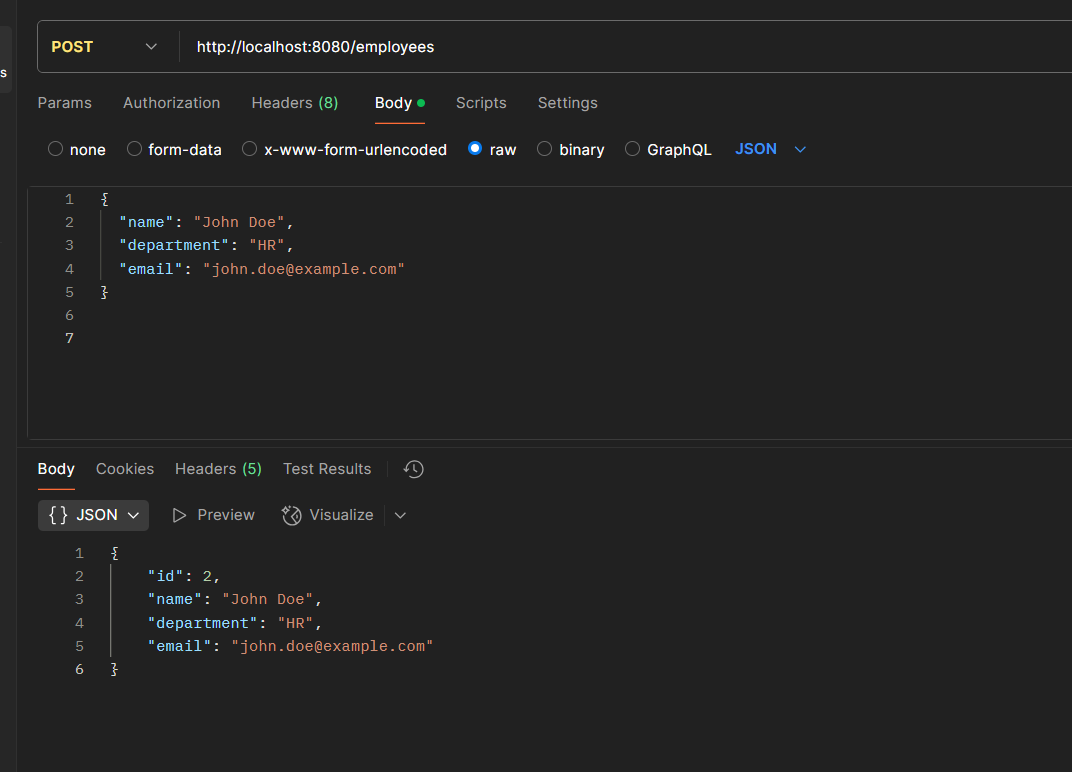
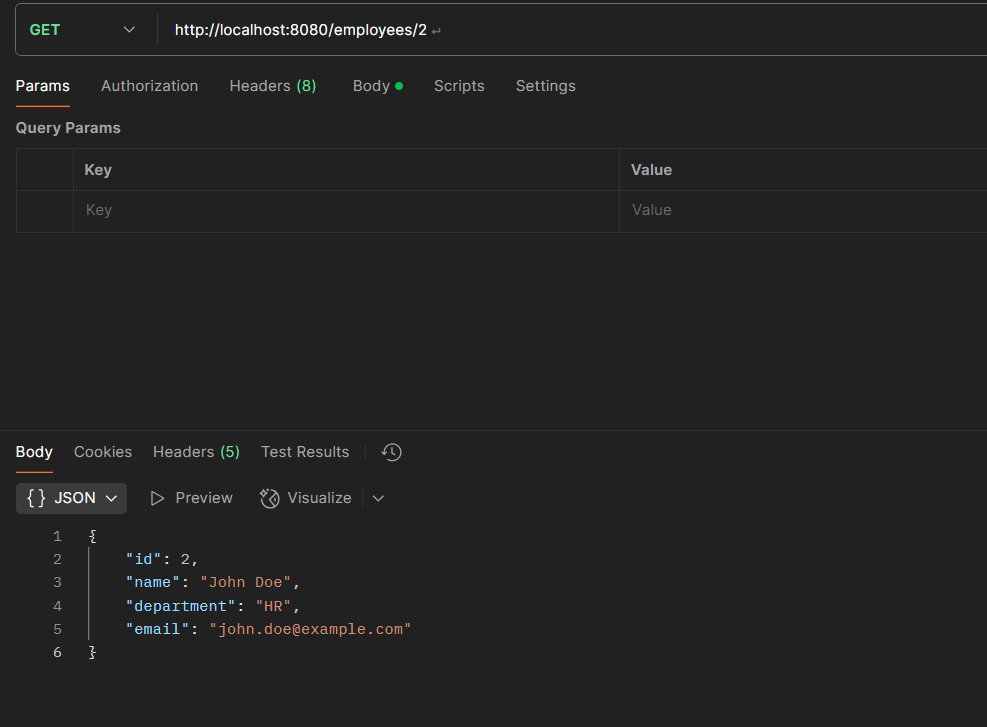
    private String name;

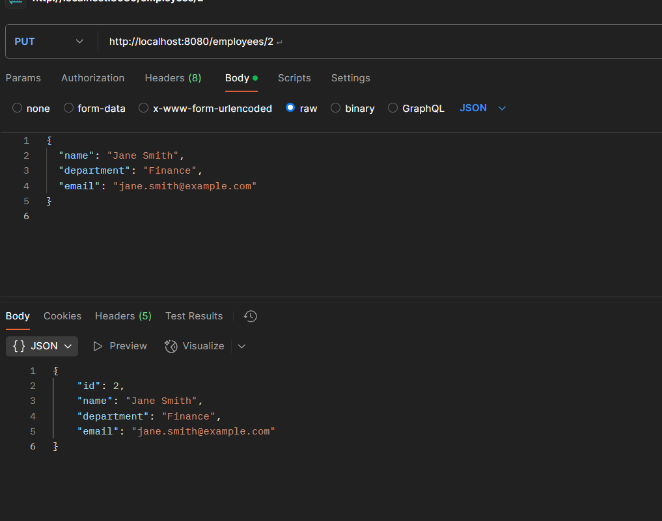
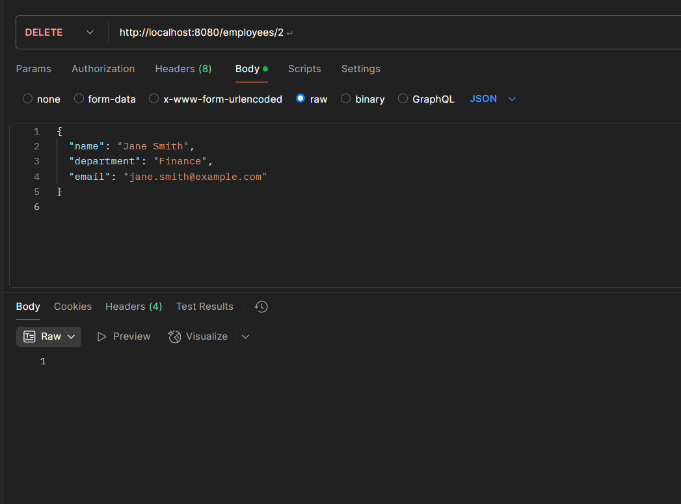
    private String department;

    private String email;

}

**Output:**

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

**Code:**

**Employee.java:**

package com.company.entity;

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Department getDepartment() {

return department;

}

public void setDepartment(Department department) {

this.department = department;

}

}

**Department.java:**

package com.company.entity;

import jakarta.persistence.\*;

import java.util.List;

@Entity

@Table(name = "departments")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

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

private List<Employee> employees;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<Employee> getEmployees() {

return employees;

}

public void setEmployees(List<Employee> employees) {

this.employees = employees;

}

}

**Output:**



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

**Overview of Spring Data Repositories**

Spring Data JPA provides ready-to-use interfaces like JpaRepository, which helps perform CRUD operations without writing boilerplate code.

**Benefits:**

* No need to write SQL or JPQL for basic operations.
* Supports pagination, sorting, and derived queries.
* Integrates easily with Spring Boot for fast development

**Code:**

**EmployeeRepository.java:**

package com.company.repository;

import com.company.entity.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**DepartmentRepository.java:**

package com.company.repository;

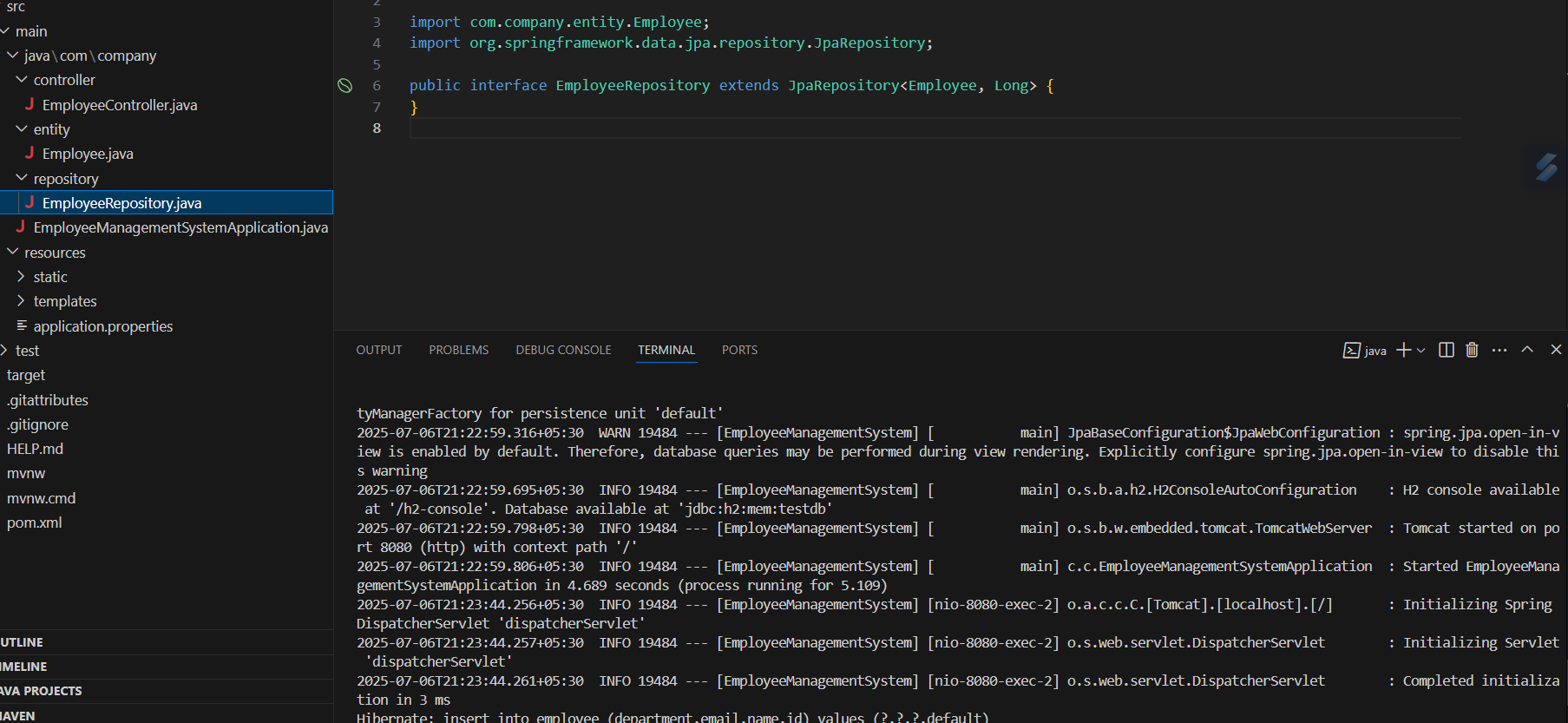
import com.company.entity.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**Output:**



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

**Code:**

**DepartmentController.java:**

package com.company.controller;

import com.company.entity.Department;

import com.company.repository.DepartmentRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/departments")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

@PostMapping

public Department addDepartment(@RequestBody Department department) {

return departmentRepository.save(department);

}

@GetMapping

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable Long id) {

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

}

@PutMapping("/{id}")

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

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

if (department != null) {

department.setName(deptDetails.getName());

return departmentRepository.save(department);

}

return null;

}

@DeleteMapping("/{id}")

public void deleteDepartment(@PathVariable Long id) {

departmentRepository.deleteById(id);

}

}

**EmployeeController.java:**

package com.company.controller;

import com.company.entity.Employee;

import com.company.repository.EmployeeRepository;

import com.company.repository.DepartmentRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private DepartmentRepository departmentRepository;

@PostMapping

public Employee addEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable Long id) {

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

}

@PutMapping("/{id}")

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

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

if (employee != null) {

employee.setName(empDetails.getName());

employee.setEmail(empDetails.getEmail());

employee.setDepartment(empDetails.getDepartment());

return employeeRepository.save(employee);

}

return null;

}

@DeleteMapping("/{id}")

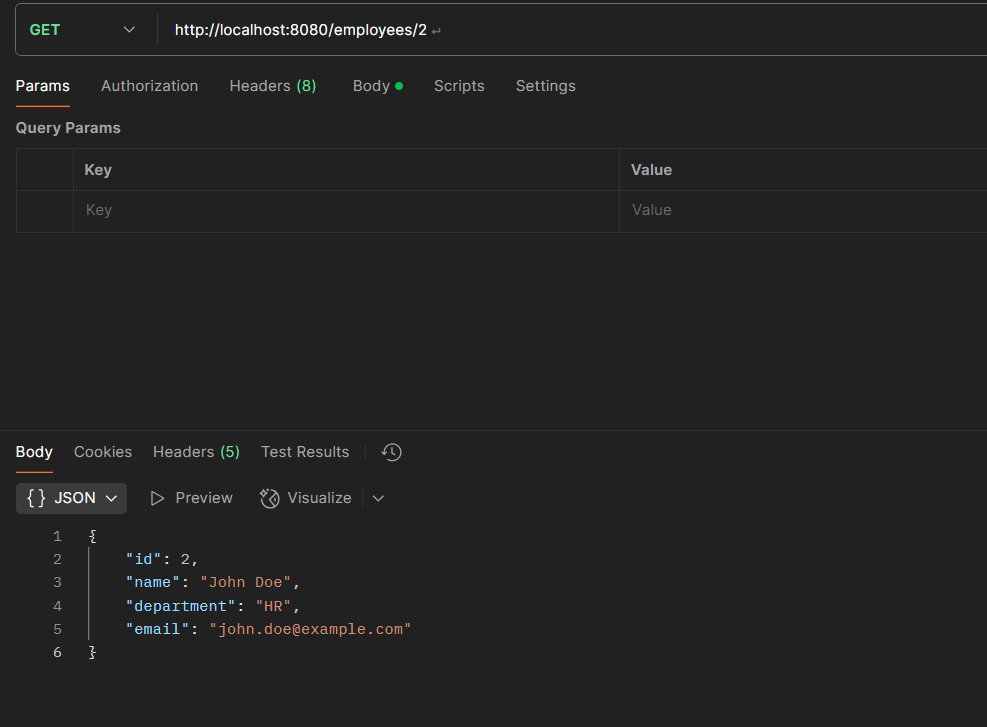
public void deleteEmployee(@PathVariable Long id) {

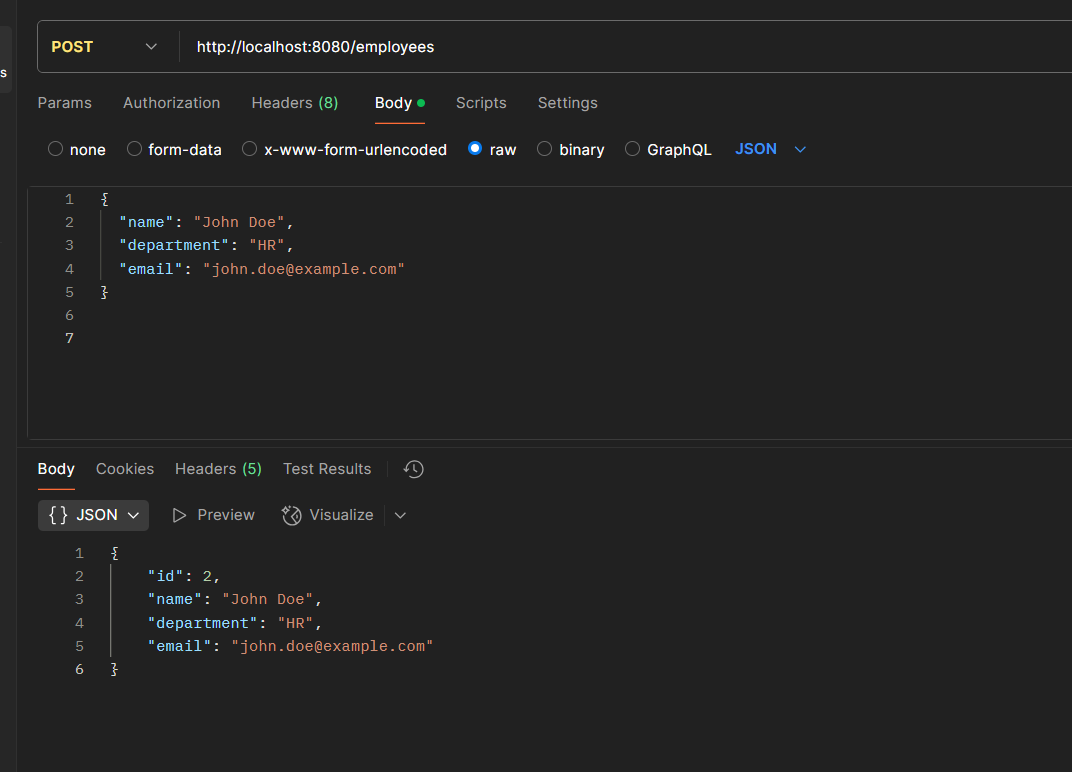
employeeRepository.deleteById(id);

}

}

**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")

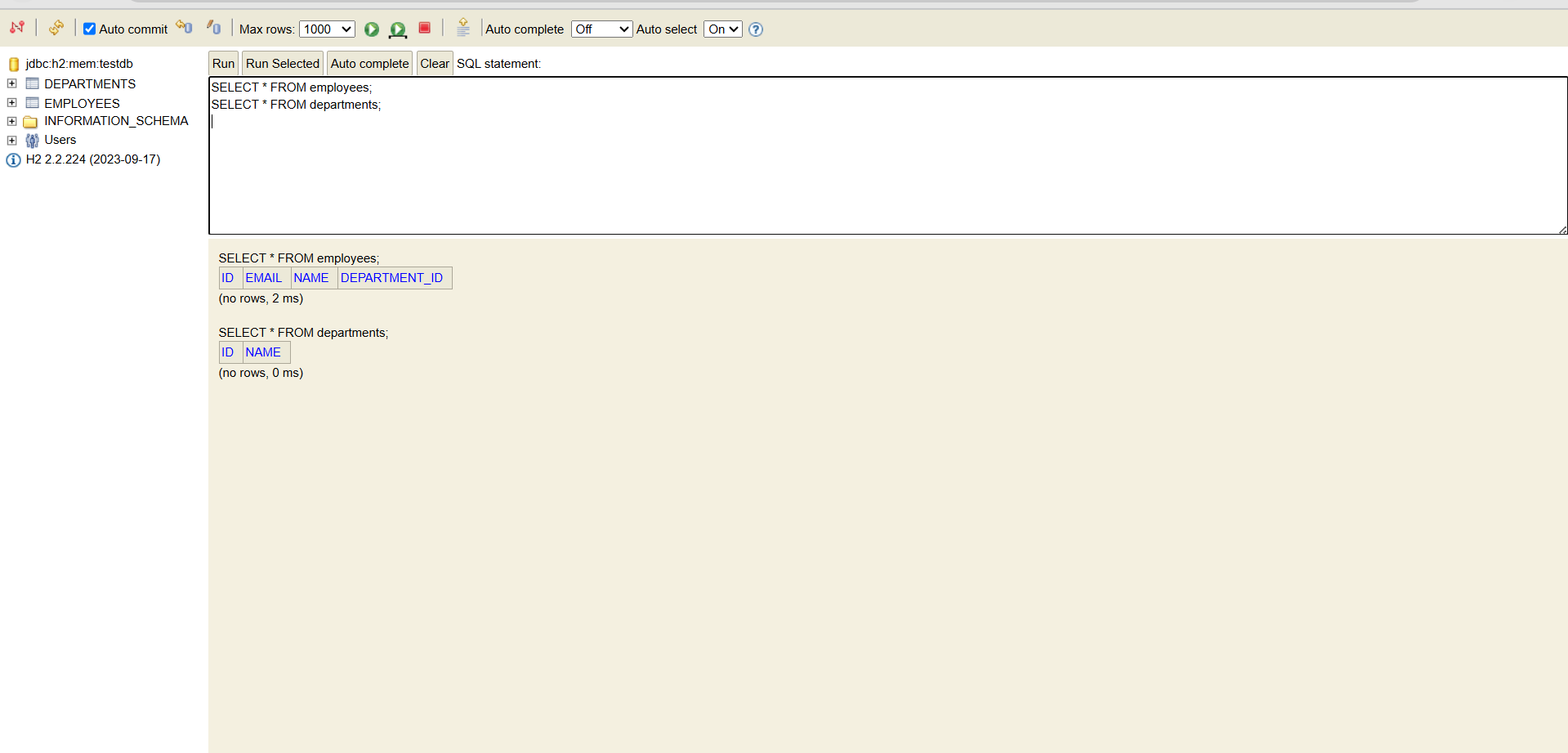
    List<Employee> findByEmail(@Param("email") String email);

    @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 EmployeeController {

    @Autowired

    private EmployeeRepository employeeRepository;

    @GetMapping

    public List<Employee> getAllEmployees() {

        return employeeRepository.findAll();

    }

    @GetMapping("/{id}")

    public Employee getEmployeeById(@PathVariable Long id) {

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

    }

    @PostMapping

    public Employee createEmployee(@RequestBody Employee employee) {

        return employeeRepository.save(employee);

    }

    @PutMapping("/{id}")

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

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

        if (employee != null) {

            employee.setName(employeeDetails.getName());

            employee.setEmail(employeeDetails.getEmail());

            employee.setDepartment(employeeDetails.getDepartment());

            return employeeRepository.save(employee);

        }

        return null;

    }

    @DeleteMapping("/{id}")

    public void deleteEmployee(@PathVariable Long id) {

        employeeRepository.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]);

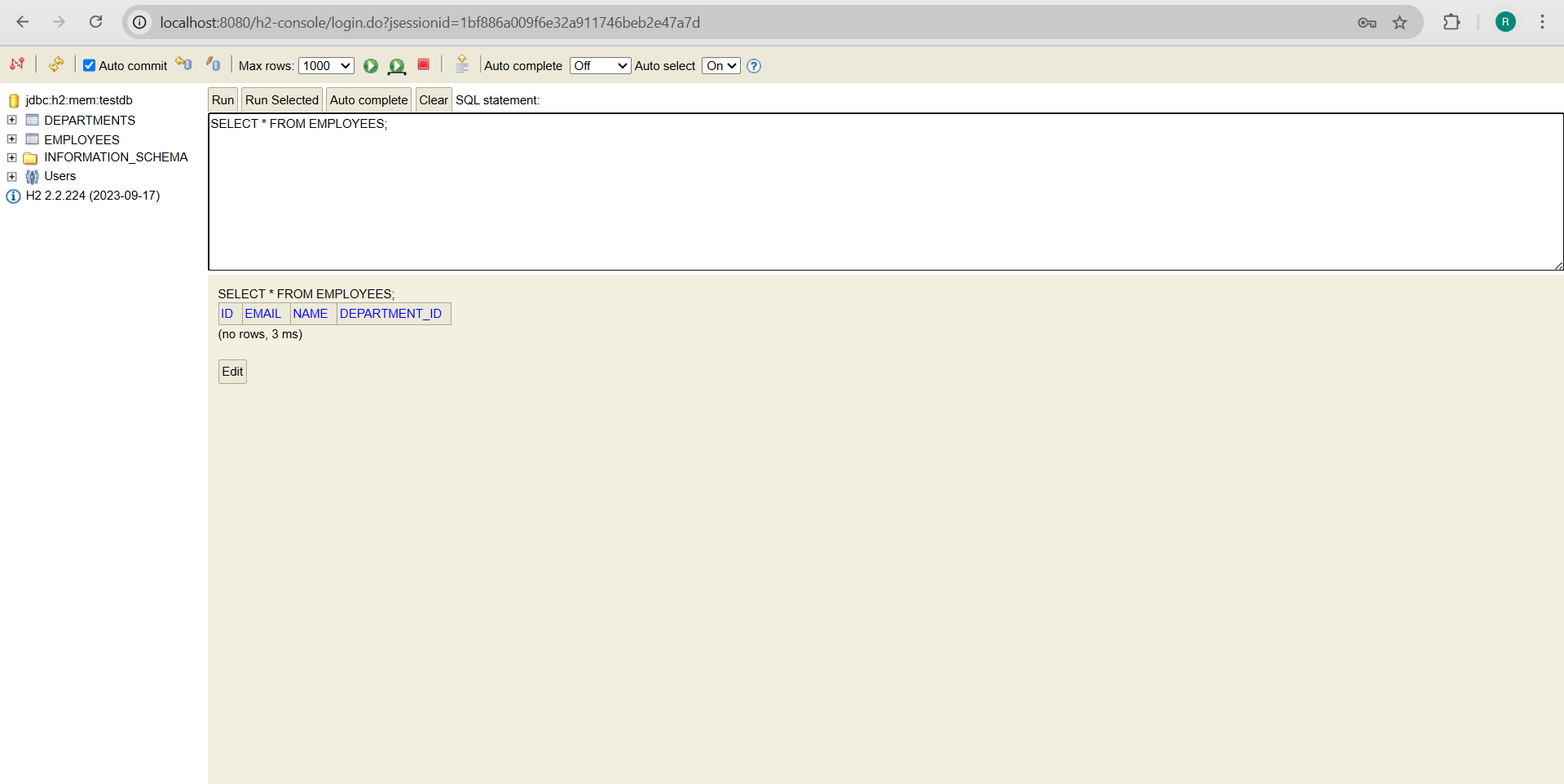
        Pageable pageable = PageRequest.of(page, size, Sort.by(sortDirection, sort[0]));

        return employeeRepository.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;

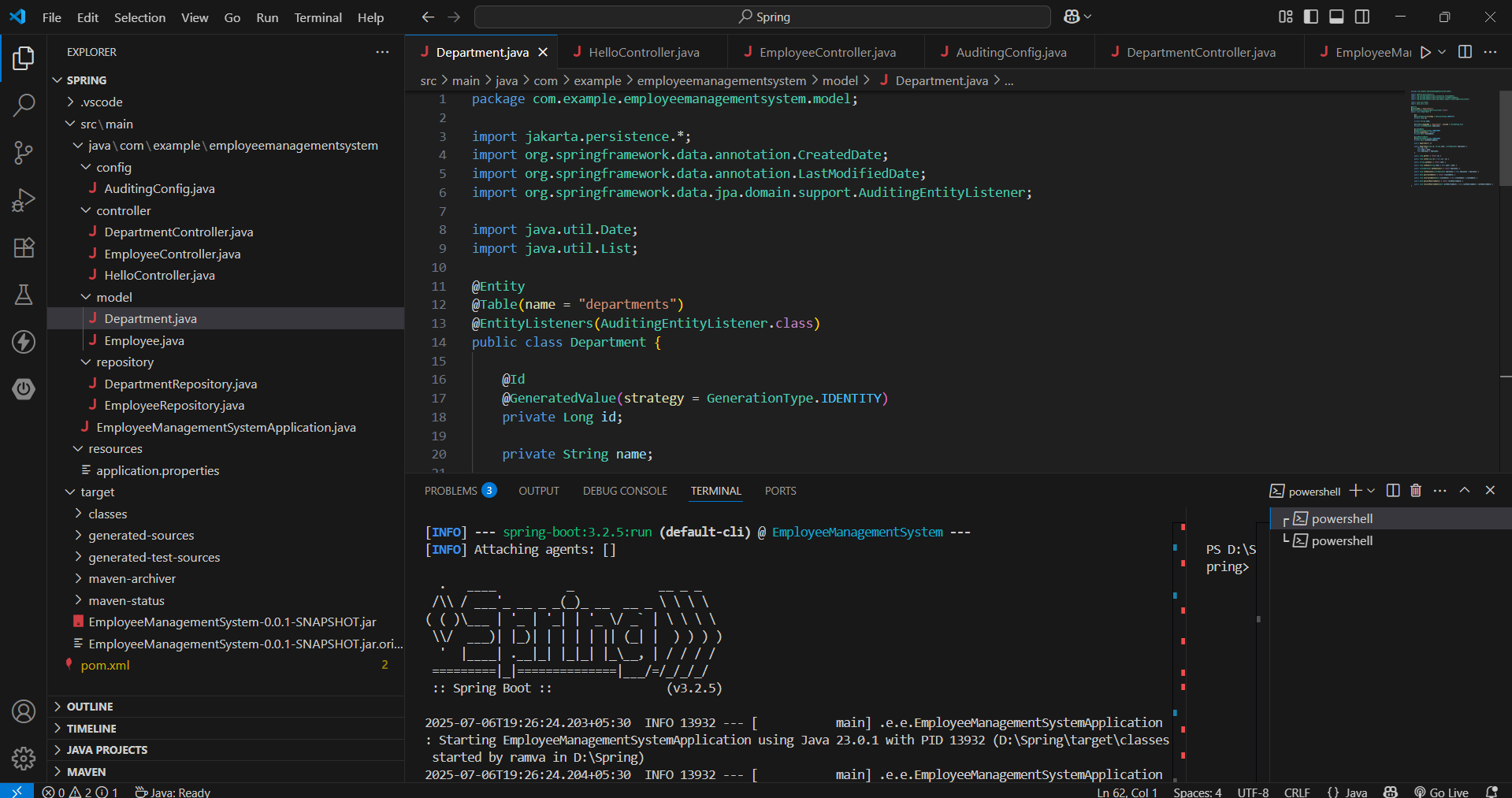
@Configuration

@EnableJpaAuditing

public class AuditingConfig {

}

**OUTPUT:**

****

**Exercise 8: Employee Management System - Creating Projections**

**Code:**

**EmployeeNameAndDept:**

package com.company.projection;

public interface EmployeeNameAndDept {

    String getName();

    String getEmail();

    DepartmentInfo getDepartment();

    interface DepartmentInfo {

        String getName();

    }

}

**EmployeeDTO:**

 package com.company.dto;

public class EmployeeDTO {

    private String name;

    private String departmentName;

    public EmployeeDTO(String name, String departmentName) {

        this.name = name;

        this.departmentName = departmentName;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getDepartmentName() {

        return departmentName;

    }

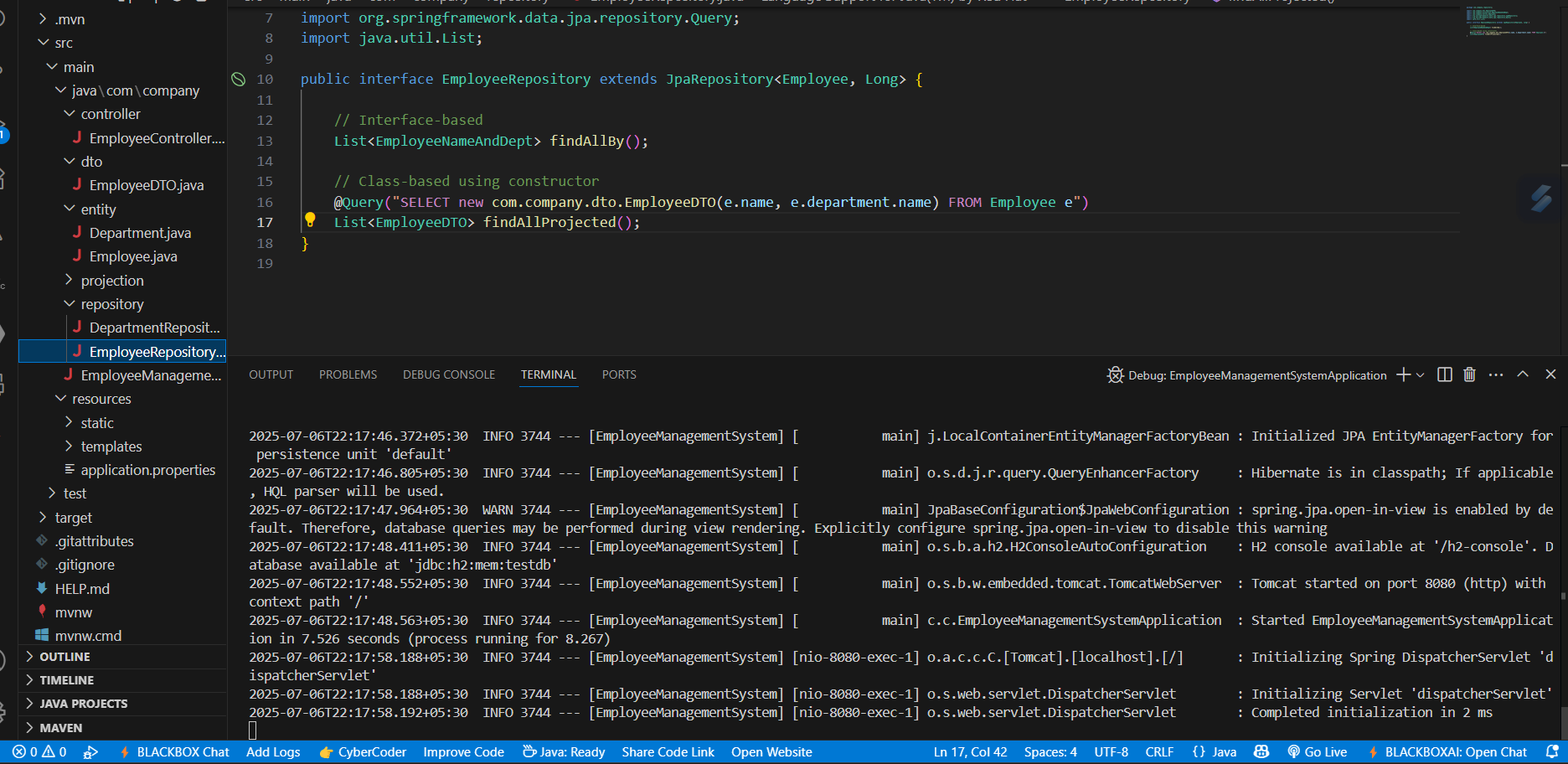
    public void setDepartmentName(String departmentName) {

        this.departmentName = departmentName;

    }

}

**Output:**

****

**Exercise 9: Employee Management System - Customizing Data Source Configuration**

**Code:**

**PrimaryDataSourceConfig.java:**

package com.company.config;

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

import org.springframework.boot.autoconfigure.orm.jpa.JpaProperties;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.jdbc.DataSourceBuilder;

import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;

import org.springframework.context.annotation.\*;

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

import org.springframework.orm.jpa.\*;

import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;

import org.springframework.transaction.PlatformTransactionManager;

import jakarta.persistence.EntityManagerFactory;

import javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

    basePackages = "com.company.repository.primary",

    entityManagerFactoryRef = "primaryEntityManagerFactory",

    transactionManagerRef = "primaryTransactionManager"

)

public class PrimaryDataSourceConfig {

    @Primary

    @Bean

    @ConfigurationProperties("spring.datasource")

    public DataSource primaryDataSource() {

        return DataSourceBuilder.create().build();

    }

    @Primary

    @Bean

    public LocalContainerEntityManagerFactoryBean primaryEntityManagerFactory(

            @Qualifier("entityManagerFactoryBuilder") EntityManagerFactoryBuilder builder) {

        return builder

                .dataSource(primaryDataSource())

                .packages("com.company.entity.primary")

                .persistenceUnit("primary")

                .build();

    }

    @Primary

    @Bean

    public PlatformTransactionManager primaryTransactionManager(

            @Qualifier("primaryEntityManagerFactory") EntityManagerFactory emf) {

        return new JpaTransactionManager(emf);

    }

    @Primary

    @Bean

    public EntityManagerFactoryBuilder entityManagerFactoryBuilder(

            JpaVendorAdapter jpaVendorAdapter, JpaProperties jpaProperties) {

        return new EntityManagerFactoryBuilder(jpaVendorAdapter, jpaProperties.getProperties(), null);

    }

    @Primary

    @Bean

    public JpaVendorAdapter jpaVendorAdapter() {

        return new HibernateJpaVendorAdapter();

    }

}

**SecondaryDataSourceConfig.java:**

package com.company.config;

import org.springframework.boot.autoconfigure.orm.jpa.JpaProperties;

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

import org.springframework.boot.autoconfigure.jdbc.DataSourceProperties;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder;

import org.springframework.context.annotation.\*;

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

import org.springframework.orm.jpa.\*;

import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;

import org.springframework.transaction.PlatformTransactionManager;

import jakarta.persistence.EntityManagerFactory;

import javax.sql.DataSource;

@Configuration

@EnableJpaRepositories(

    basePackages = "com.company.repository.secondary",

    entityManagerFactoryRef = "secondaryEntityManagerFactory",

    transactionManagerRef = "secondaryTransactionManager"

)

public class SecondaryDataSourceConfig {

    @Bean

    @ConfigurationProperties("secondary.datasource")

    public DataSourceProperties secondaryDataSourceProperties() {

        return new DataSourceProperties();

    }

    @Bean

    public DataSource secondaryDataSource() {

        return secondaryDataSourceProperties().initializeDataSourceBuilder().build();

    }

    @Bean

    public LocalContainerEntityManagerFactoryBean secondaryEntityManagerFactory(

            @Qualifier("secondaryEntityManagerFactoryBuilder") EntityManagerFactoryBuilder builder) {

        return builder

                .dataSource(secondaryDataSource())

                .packages("com.company.entity.secondary")

                .persistenceUnit("secondary")

                .build();

    }

    @Bean

    public PlatformTransactionManager secondaryTransactionManager(

            @Qualifier("secondaryEntityManagerFactory") EntityManagerFactory emf) {

        return new JpaTransactionManager(emf);

    }

    @Bean

    public EntityManagerFactoryBuilder secondaryEntityManagerFactoryBuilder(

            JpaVendorAdapter jpaVendorAdapter, JpaProperties jpaProperties) {

        return new EntityManagerFactoryBuilder(jpaVendorAdapter, jpaProperties.getProperties(), null);

    }

    @Bean

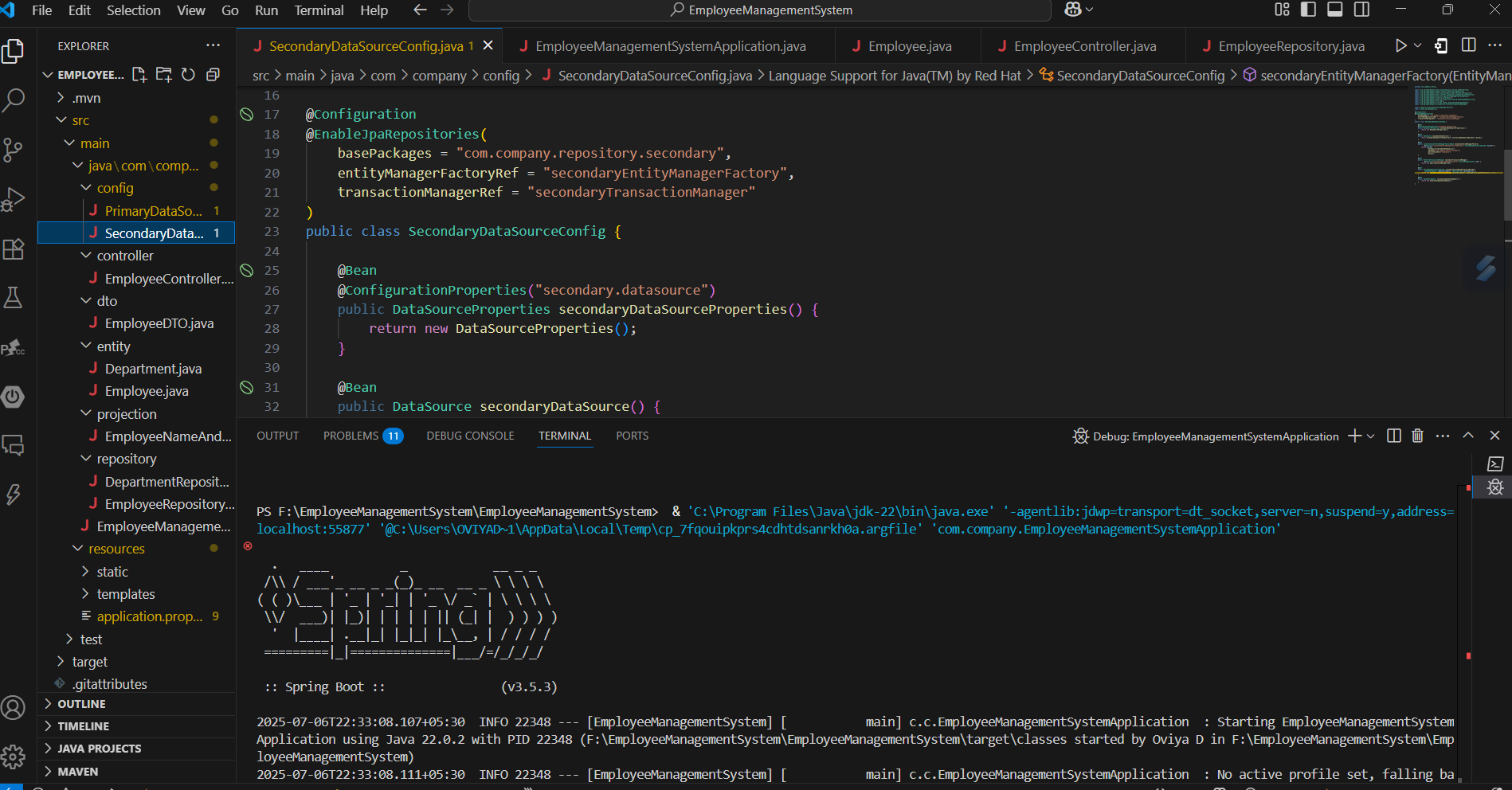
    public JpaVendorAdapter secondaryJpaVendorAdapter() {

        return new HibernateJpaVendorAdapter();

    }

}

**Output:**

****

**Exercise 10: Employee Management System - Hibernate-Specific Features**

**Code:**

**EmployeeBatchService.java:**package com.company.service;

import com.company.entity.Employee;

import jakarta.persistence.EntityManager;

import jakarta.persistence.PersistenceContext;

import jakarta.transaction.Transactional;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeBatchService {

@PersistenceContext

private EntityManager entityManager;

@Transactional

public void saveEmployeesInBatch(List<Employee> employees) {

for (int i = 0; i < employees.size(); i++) {

entityManager.persist(employees.get(i));

if (i % 20 == 0) {

entityManager.flush();

entityManager.clear();

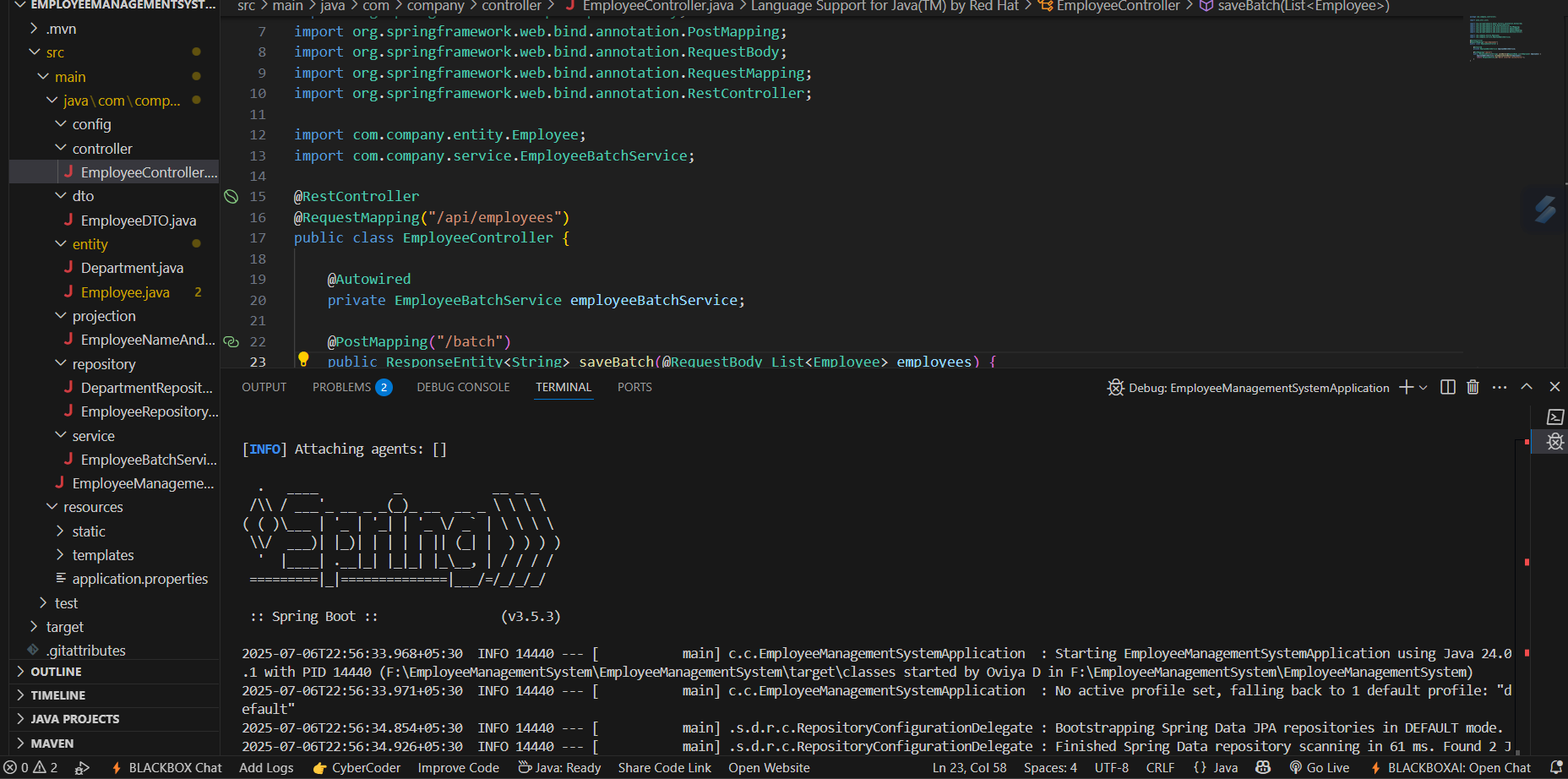
}

}

}

}

**Output:**

****