**WEEK-3**

**SPRING DATA JPA AND HIBERNATE HANDS ONS**

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

**Code:**

**EmployeeManagementSystemApplication.java**  
package com.example.employeemanagementsystem;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementSystemApplication {

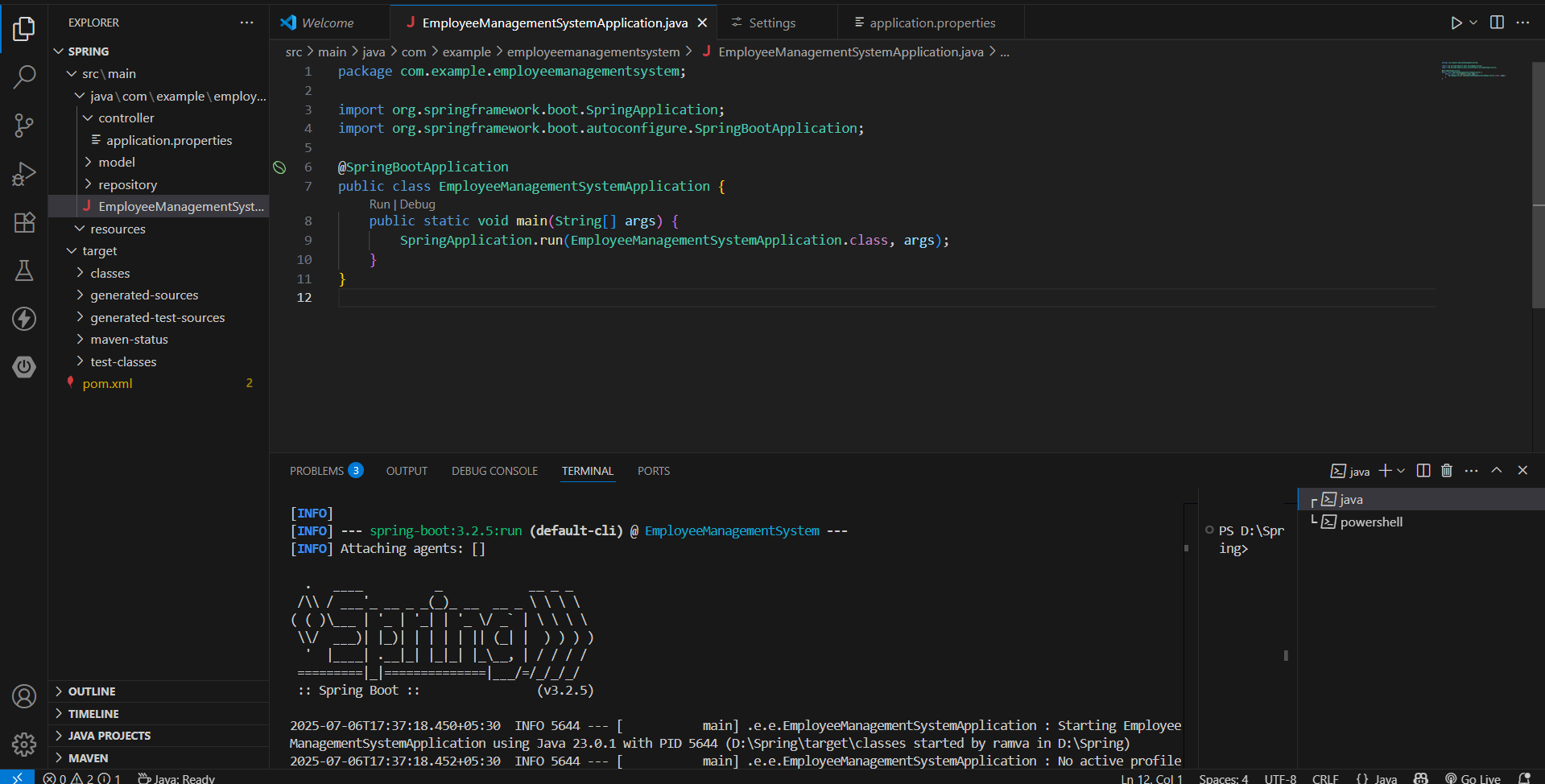
public static void main(String[] args) {

SpringApplication.run(EmployeeManagementSystemApplication.class, args);

}

}

**Output:**



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

**Code:**

**WelcomeController.java:**

package com.example.employeemanagementsystem.controller;

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

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

@RestController

public class WelcomeController {

@GetMapping("/")

public String welcomeMessage() {

return "Employee Management System is running!";

}

}

**Dept.java:**  
package com.example.employeemanagementsystem.model;

import jakarta.persistence.\*;

import lombok.\*;

import java.util.List;

@Entity

@Table(name = "departments")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Dept {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long deptId;

private String deptName;

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

private List<Staff> staffList;

}

**Staff.java:**

package com.example.employeemanagementsystem.model;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Table(name = "employees")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Staff {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long empId;

private String empName;

private String empEmail;

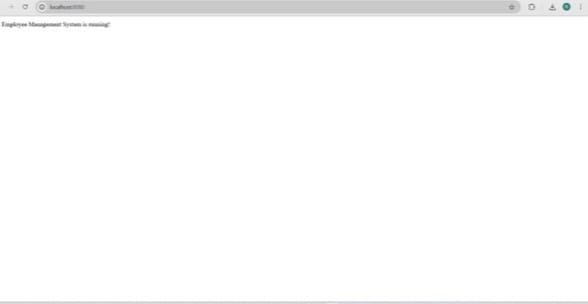
@ManyToOne

@JoinColumn(name = "department\_id")

private Dept department;

}

**Output:**



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

**Code:**

**DeptRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Dept;

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

public interface DeptRepository extends JpaRepository<Dept, Long> {

Dept findByDeptName(String deptName);

}

**StaffRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Staff;

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 StaffRepository extends JpaRepository<Staff, Long> {

List<Staff> findByEmpName(String empName);

List<Staff> findByDepartment\_DeptName(String deptName);

@Query("SELECT s FROM Staff s WHERE s.empEmail = :email")

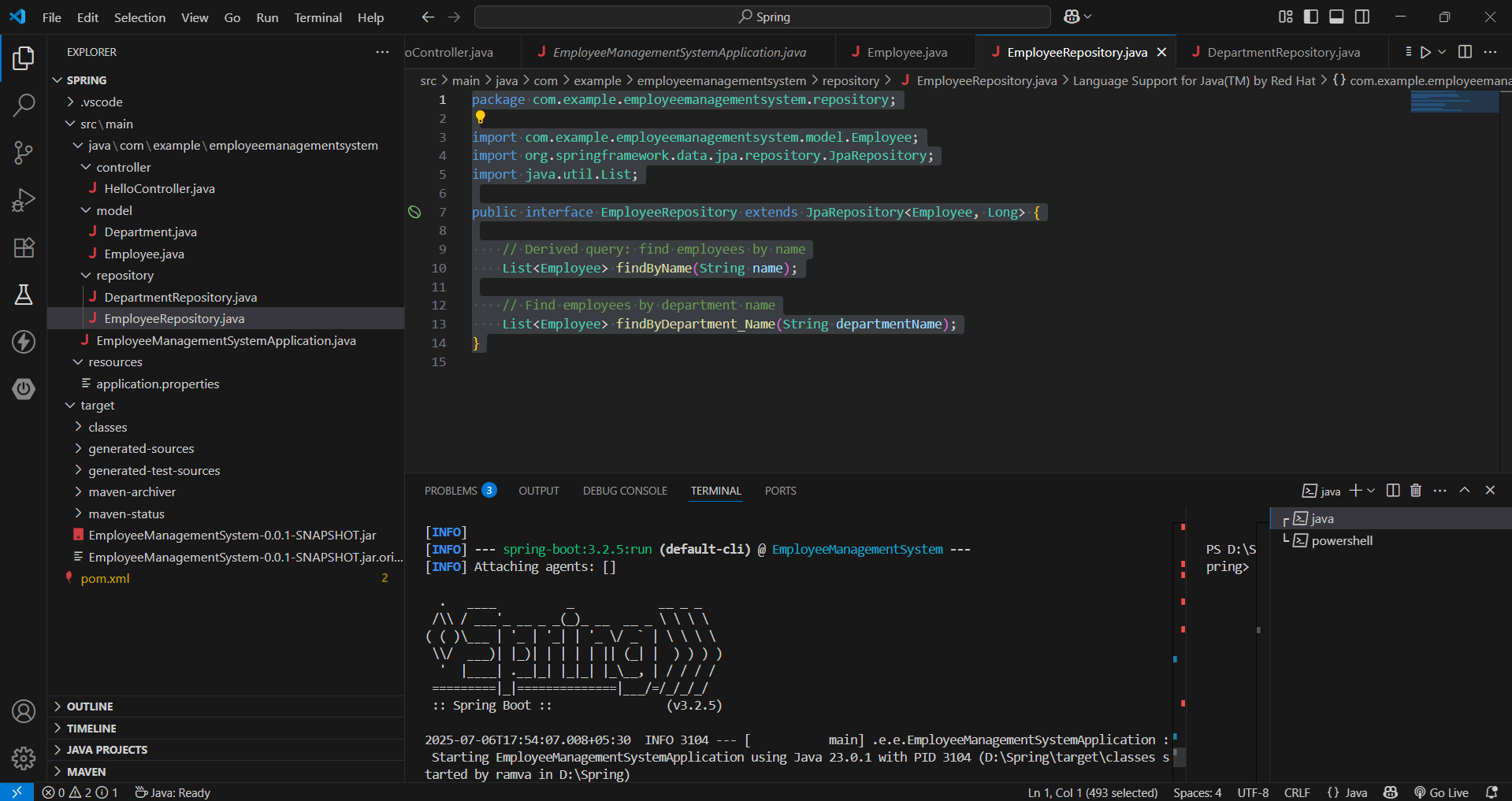
List<Staff> findByEmail(@Param("email") String email);

@Query("SELECT s.empEmail FROM Staff s")

List<String> getAllEmails();

}

**Output:**

****

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

**Code:**

**DeptController.java:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Dept;

import com.example.employeemanagementsystem.repository.DeptRepository;

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 DeptRepository deptRepo;

@GetMapping

public List<Dept> fetchAllDepartments() {

return deptRepo.findAll();

}

@GetMapping("/{id}")

public Dept fetchDepartmentById(@PathVariable Long id) {

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

}

@PostMapping

public Dept addDepartment(@RequestBody Dept dept) {

return deptRepo.save(dept);

}

@PutMapping("/{id}")

public Dept modifyDepartment(@PathVariable Long id, @RequestBody Dept deptDetails) {

Dept dept = deptRepo.findById(id).orElse(null);

if (dept != null) {

dept.setDeptName(deptDetails.getDeptName());

return deptRepo.save(dept);

}

return null;

}

@DeleteMapping("/{id}")

public void removeDepartment(@PathVariable Long id) {

deptRepo.deleteById(id);

}

}

**StaffController.java:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Staff;

import com.example.employeemanagementsystem.repository.StaffRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/employees")

public class StaffController {

@Autowired

private StaffRepository staffRepo;

@GetMapping

public List<Staff> fetchAllEmployees() {

return staffRepo.findAll();

}

@GetMapping("/{id}")

public Staff fetchEmployeeById(@PathVariable Long id) {

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

}

@PostMapping

public Staff addEmployee(@RequestBody Staff staff) {

return staffRepo.save(staff);

}

@PutMapping("/{id}")

public Staff modifyEmployee(@PathVariable Long id, @RequestBody Staff staffDetails) {

Staff emp = staffRepo.findById(id).orElse(null);

if (emp != null) {

emp.setEmpName(staffDetails.getEmpName());

emp.setEmpEmail(staffDetails.getEmpEmail());

emp.setDepartment(staffDetails.getDepartment());

return staffRepo.save(emp);

}

return null;

}

@DeleteMapping("/{id}")

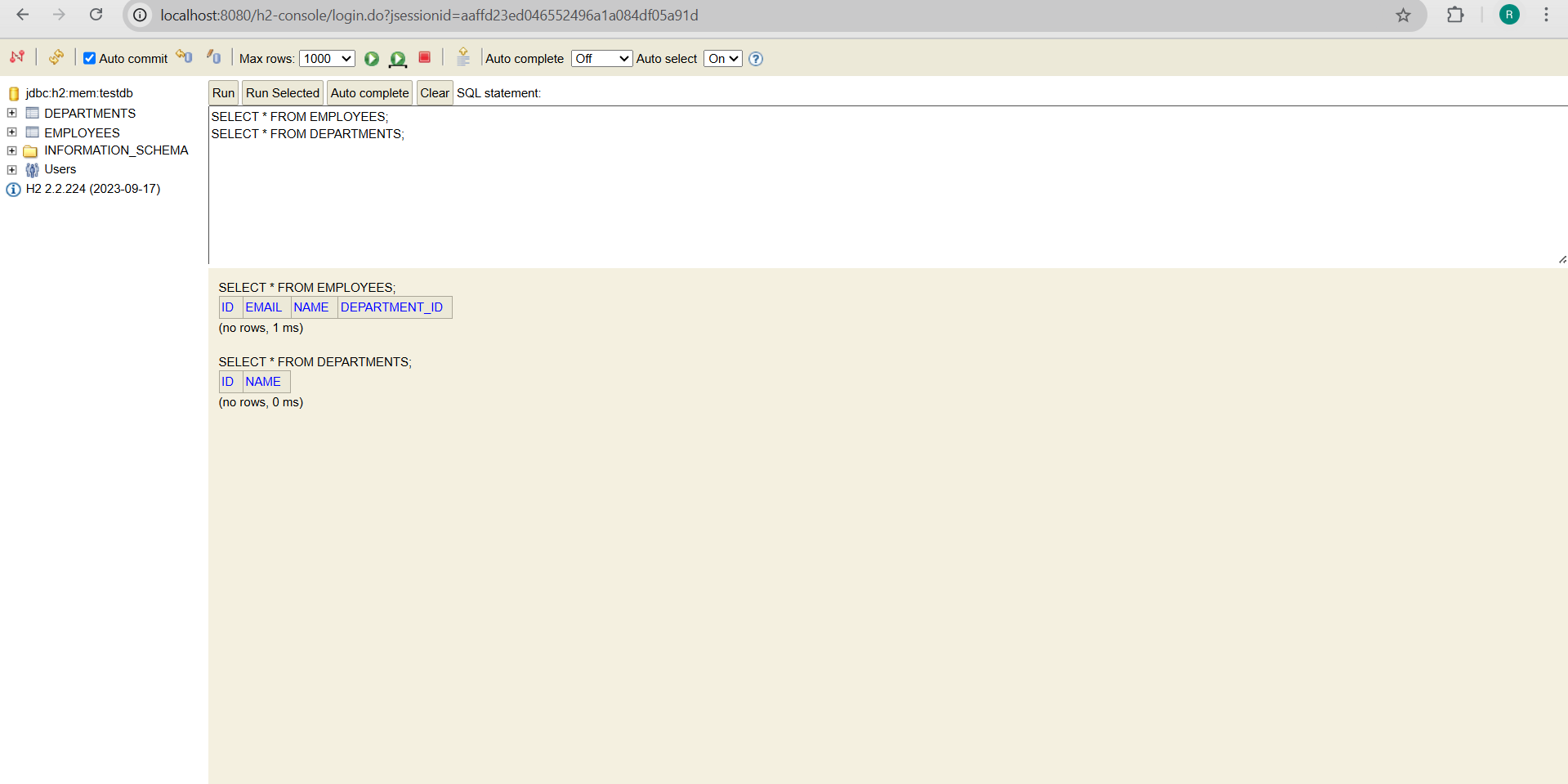
public void removeEmployee(@PathVariable Long id) {

staffRepo.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")

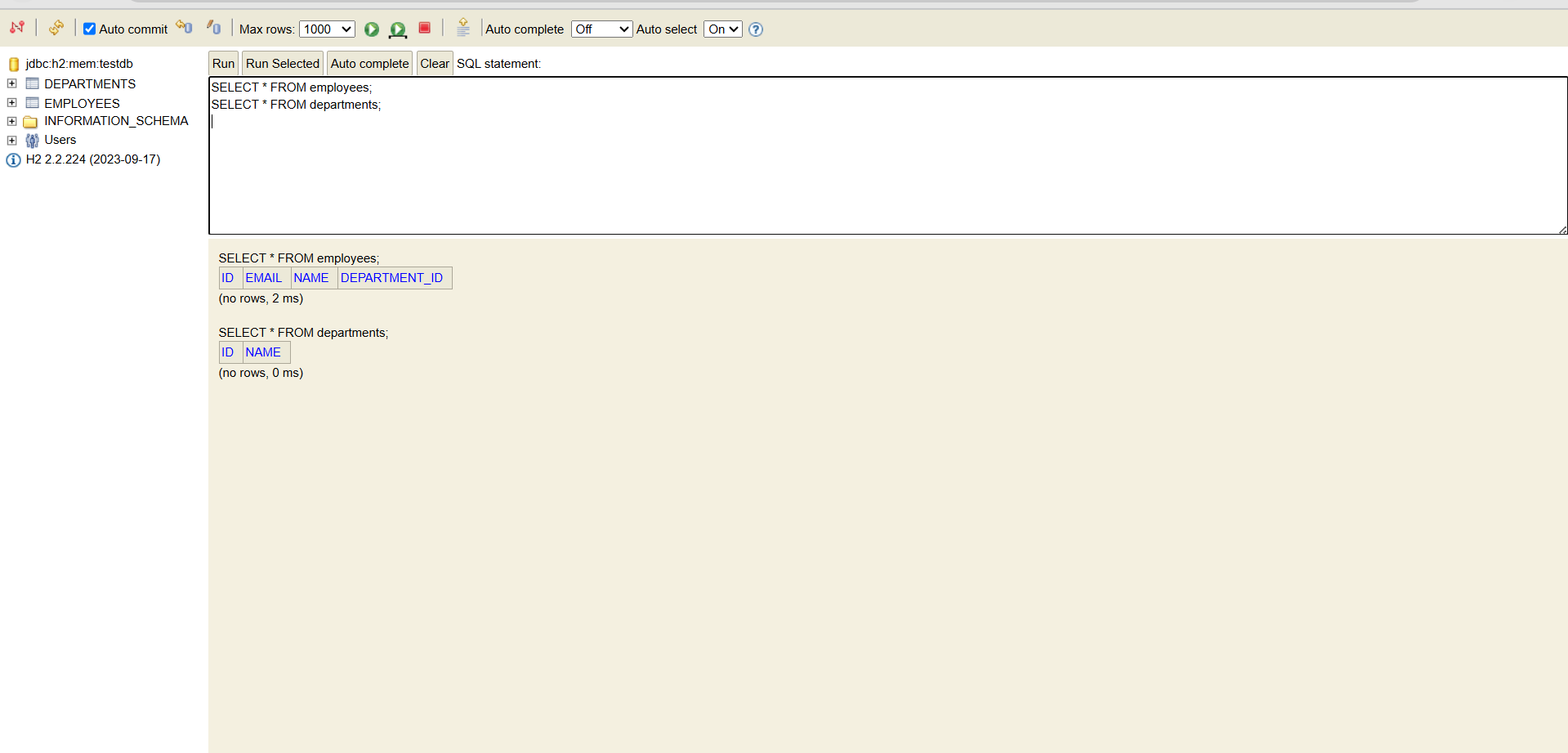
    List<Employee> findByEmail(@Param("email") String email);

    @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]);

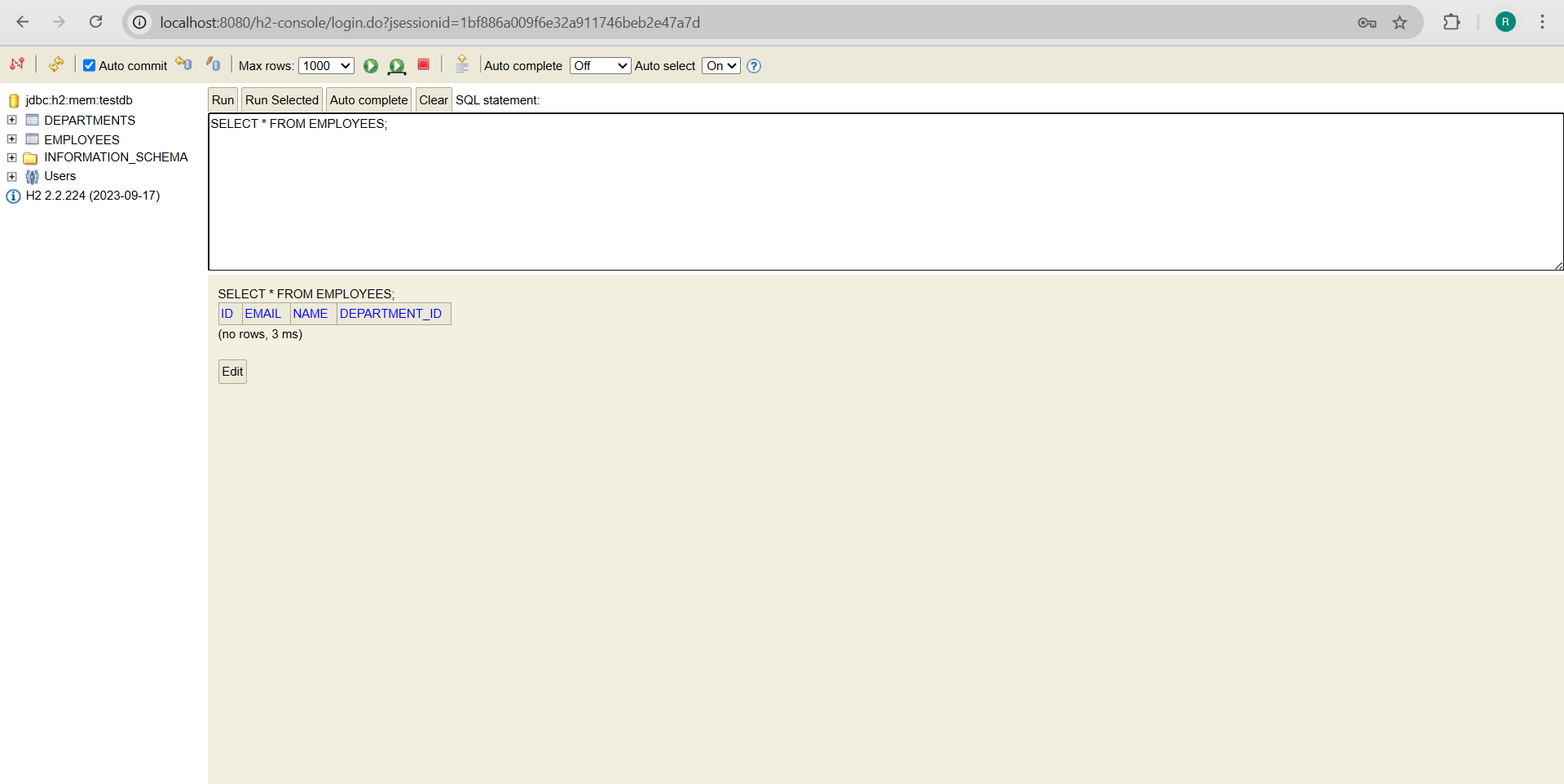
        Pageable pageable = PageRequest.of(page, size, Sort.by(sortDirection, sort[0]));

        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;

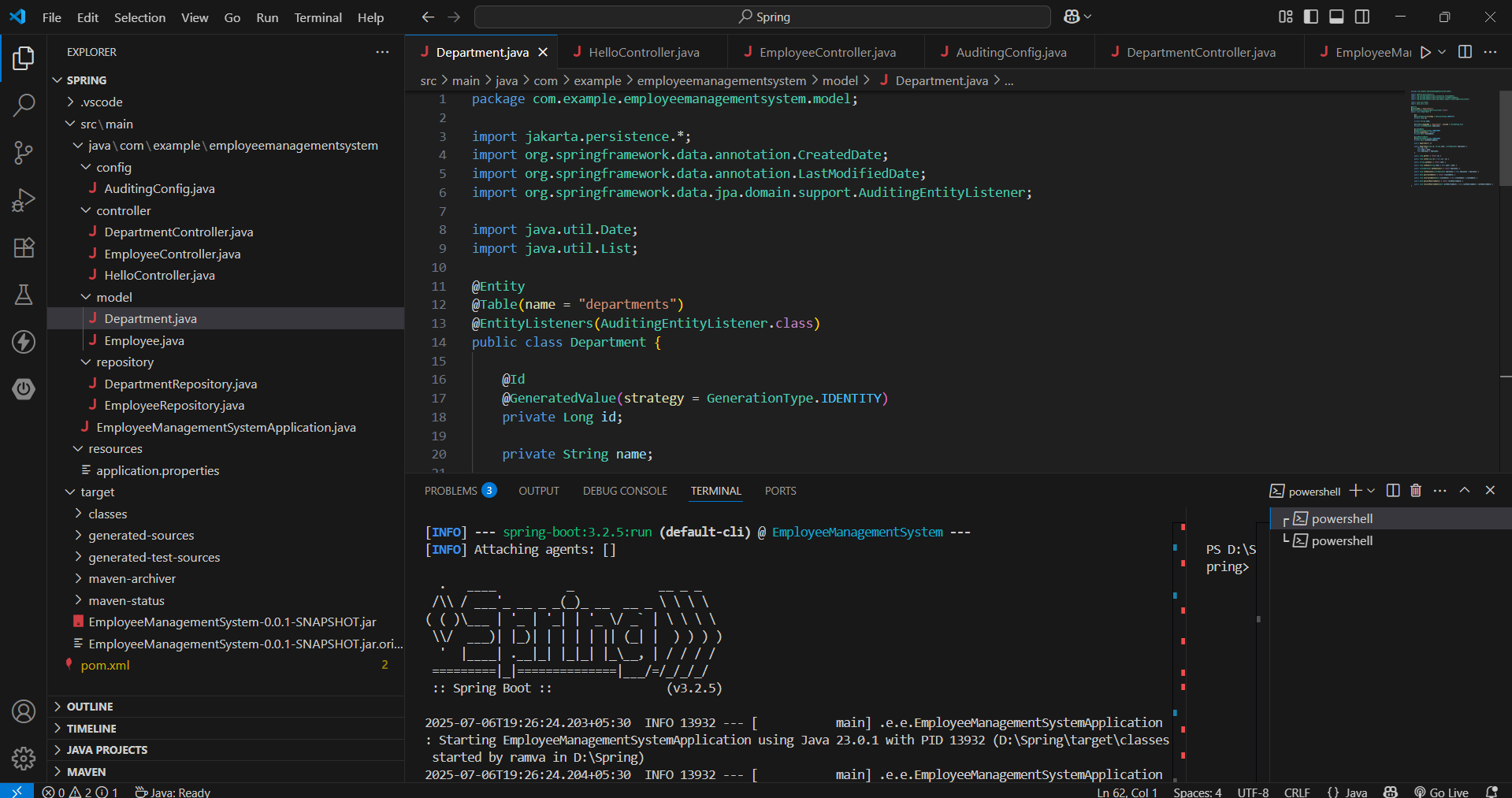
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@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;

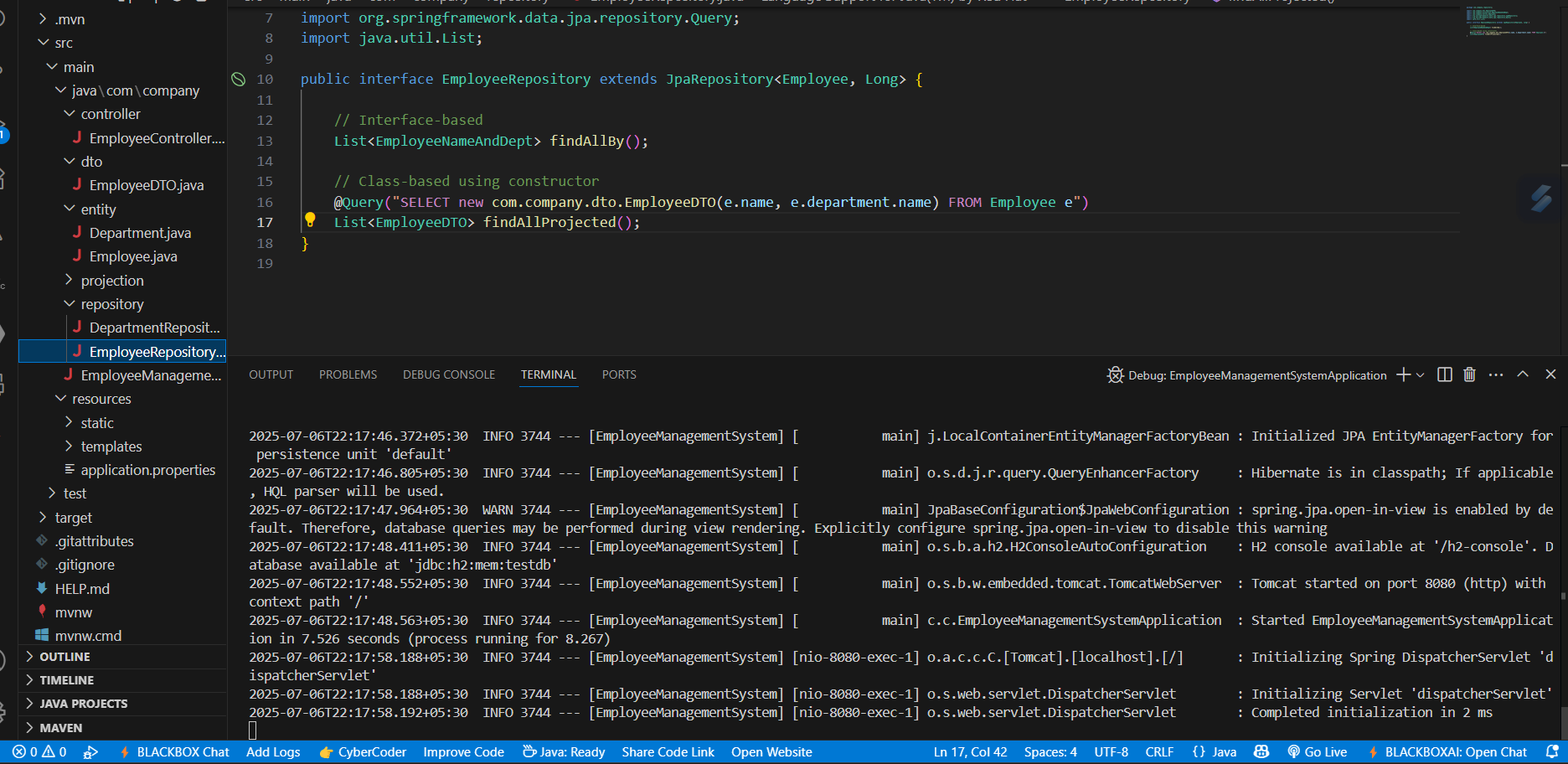
    }

    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

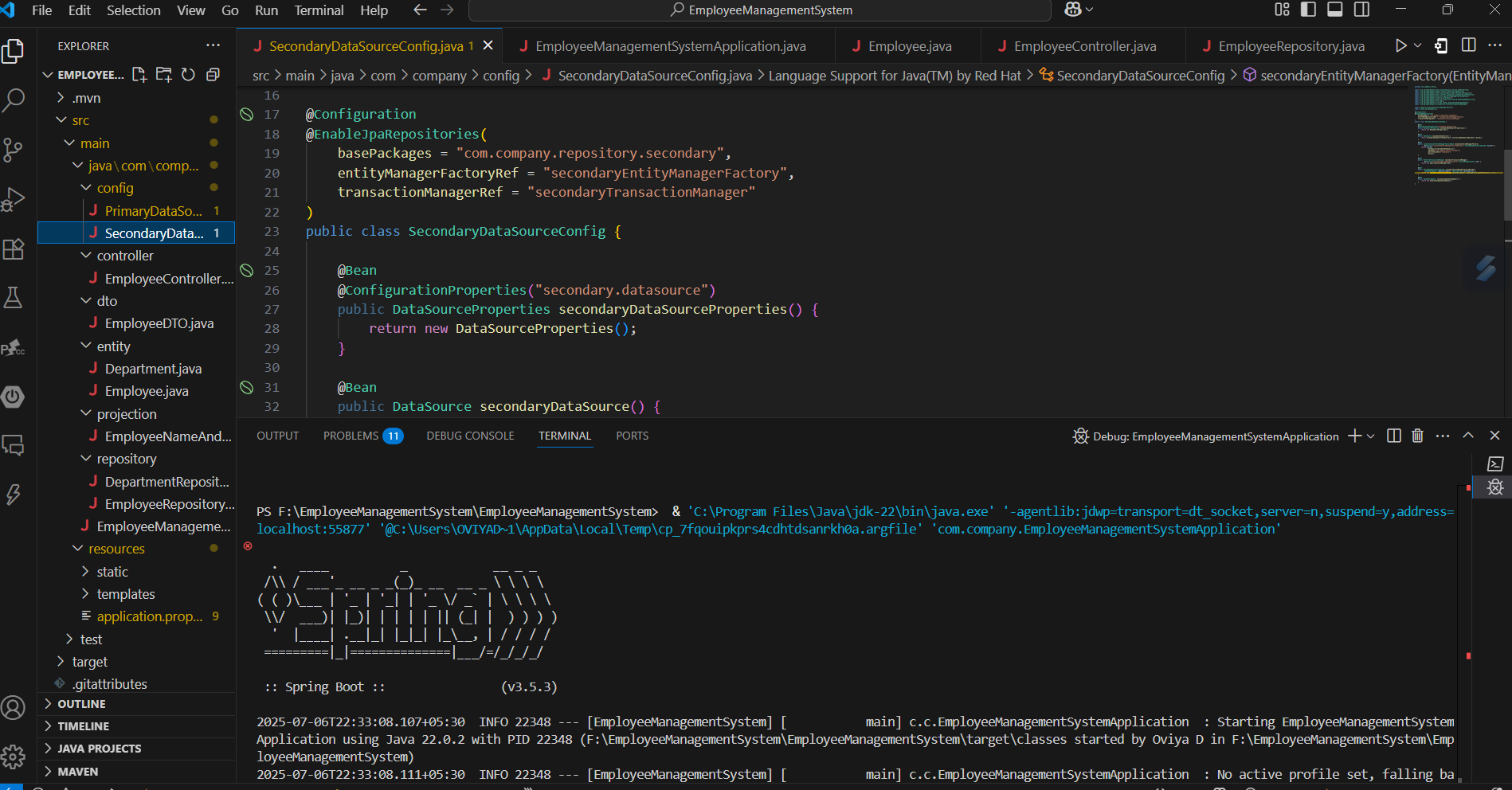
    public JpaVendorAdapter secondaryJpaVendorAdapter() {

        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:**

