**WEEK – 03**

**Spring Data JPA with Spring Boot, Hibernate**

**Superset ID: 6262264**

**Exercise 1:**

**Spring Data JPA - Quick Example**

**SOLUTION:**

* The Country.java class is defined as an entity that maps to the country table in the database. It uses JPA annotations to bind the fields code and name to their respective columns.
* The CountryRepository.java interface extends JpaRepository, providing ready-made CRUD operations for the Country entity without requiring any extra implementation.
* The CountryService.java class serves as the service layer where the application’s business logic resides. It injects the CountryRepository and exposes methods such as getAllCountries() to retrieve data from the database.

**Below is the implemented code:**

***Country.java***

*package com.cognizant.orm\_learn.model;*

*import jakarta.persistence.Column;*

*import jakarta.persistence.Entity;*

*import jakarta.persistence.Id;*

*import jakarta.persistence.Table;*

*@Entity*

*@Table(name = "country")*

*public class Country {*

*@Id*

*@Column(name = "code")*

*private String code;*

*@Column(name = "name")*

*private String name;*

*public String getCode() {*

*return code;*

*}*

*public void setCode(String code) {*

*this.code = code;*

*}*

*public String getName() {*

*return name;*

*}*

*public void setName(String name) {*

*this.name = name;*

*}*

*@Override*

*public String toString() {*

*return "Country [code=" + code + ", name=" + name + "]";*

*}*

*}*

***CountryRepository.java***

*package com.cognizant.orm\_learn.repository;*

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

*import org.springframework.stereotype.Repository;*

*import com.cognizant.orm\_learn.model.Country;*

*@Repository*

*public interface CountryRepository extends JpaRepository<Country, String> {*

*}*

***CountryService.java***

*package com.cognizant.orm\_learn.service;*

*import java.util.List;*

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

*import org.springframework.stereotype.Service;*

*import org.springframework.transaction.annotation.Transactional;*

*import com.cognizant.orm\_learn.model.Country;*

*import com.cognizant.orm\_learn.repository.CountryRepository;*

*@Service*

*public class CountryService {*

*@Autowired*

*private CountryRepository countryRepository;*

*@Transactional*

*public List<Country> getAllCountries() {*

*return countryRepository.findAll();*

*}*

*}*

***OrmLearnApplication.java***

*package com.cognizant.orm\_learn;*

*import java.util.List;*

*import org.slf4j.Logger;*

*import org.slf4j.LoggerFactory;*

*import org.springframework.boot.SpringApplication;*

*import org.springframework.boot.autoconfigure.SpringBootApplication;*

*import org.springframework.context.ApplicationContext;*

*import com.cognizant.orm\_learn.model.Country;*

*import com.cognizant.orm\_learn.service.CountryService;*

*@SpringBootApplication*

*public class OrmLearnApplication {*

*private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);*

*private static CountryService countryService;*

*public static void main(String[] args) {*

*ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);*

*countryService = context.getBean(CountryService.class);*

*displayAllCountries();*

*}*

*private static void displayAllCountries() {*

*LOGGER.info("Start retrieving countries");*

*List<Country> countries = countryService.getAllCountries();*

*LOGGER.debug("Countries retrieved: {}", countries);*

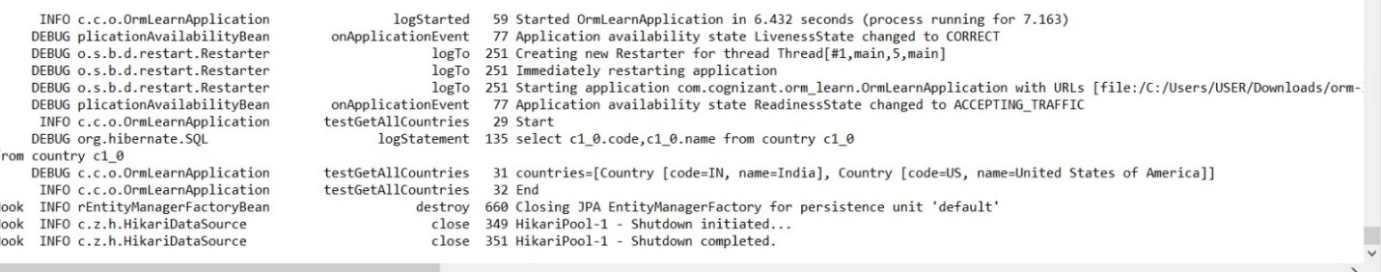
*LOGGER.info("Retrieval completed");*

*}*

*}*

***OUTPUT:***

****



**Exercise 3:**

**Difference between JPA, Hibernate and Spring Data JPA**

**SOLUTION:**

**JPA (Java Persistence API)**

* **Definition:** A *specification*—like an official rulebook—for mapping Java objects to relational database tables.
* **Origin:** Part of Java EE (currently Jakarta EE), defined under *JSR 338*.
* **Characteristics:**
  + Provides annotations and interfaces.
  + Has no built-in implementation—you need a provider (e.g., Hibernate) to actually persist data

**Hibernate**

* **Definition:** A widely adopted *Object-Relational Mapping (ORM)* framework.
* **Role:** It’s a concrete implementation of JPA.
* **Key Capabilities:**
  + Links Java classes to database tables.
  + Auto-generates SQL queries.
  + Supports caching, lazy loading, and transaction handling.

**Spring Data JPA**

* **Definition:** A Spring project that streamlines working with JPA.
* **How it works:**
  + Sits on top of JPA providers like Hibernate.
  + Does not itself implement JPA.
* **Benefits:**
  + Eliminates boilerplate DAO code.
  + Provides repository interfaces with ready-made CRUD operations.
  + Enables easy pagination, sorting, and dynamic query creation.

***Maven Dependencies (pom.xml)***

*<dependencies>*

*<dependency>*

*<groupId>org.hibernate.orm</groupId>*

*<artifactId>hibernate-core</artifactId>*

*<version>6.4.4.Final</version>*

*</dependency>*

*<dependency>*

*<groupId>com.mysql</groupId>*

*<artifactId>mysql-connector-j</artifactId>*

*<version>8.0.33</version>*

*</dependency>*

*<dependency>*

*<groupId>jakarta.persistence</groupId>*

*<artifactId>jakarta.persistence-api</artifactId>*

*<version>3.1.0</version>*

*</dependency>*

*<dependency>*

*<groupId>org.jboss.logging</groupId>*

*<artifactId>jboss-logging</artifactId>*

*<version>3.5.3.Final</version>*

*</dependency>*

*</dependencies>*

***Hibernate Configuration (hibernate.cfg.xml)***

*<!DOCTYPE hibernate-configuration PUBLIC*

*"-//Hibernate/Hibernate Configuration DTD 3.0//EN"*

*"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">*

*<hibernate-configuration>*

*<session-factory>*

*<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>*

*<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/testdb</property>*

*<property name="hibernate.connection.username">root</property>*

*<property name="hibernate.connection.password">Divya@priya135</property>*

*<property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>*

*<property name="hibernate.hbm2ddl.auto">update</property>*

*<property name="hibernate.show\_sql">true</property>*

*<mapping class="com.example.Employee"/>*

*</session-factory>*

*</hibernate-configuration>*

***Entity Class (Employee.java):***

*package com.example;*

*import jakarta.persistence.\*;*

*@Entity*

*@Table(name = "employee")*

*public class Employee {*

*@Id*

*@GeneratedValue(strategy = GenerationType.IDENTITY)*

*private int id;*

*private String name;*

*private String department;*

*// Getters and Setters*

*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 String getDepartment() { return department; }*

*public void setDepartment(String department) { this.department = department; }*

*}*

**Utility Class for SessionFactory (HibernateUtil.java)*:***

*package com.example;*

*import org.hibernate.SessionFactory;*

*import org.hibernate.cfg.Configuration;*

*public class HibernateUtil {*

*private static final SessionFactory sessionFactory;*

*static {*

*try {*

*sessionFactory = new Configuration()*

*.configure("hibernate.cfg.xml")*

*.buildSessionFactory();*

*} catch (Throwable ex) {*

*throw new ExceptionInInitializerError(ex);*

*}*

*}*

*public static SessionFactory getSessionFactory() {*

*return sessionFactory;*

*}*

*}*

***Main Application (Main.java):***

*package com.example;*

*import org.hibernate.Session;*

*import org.hibernate.Transaction;*

*public class Main {*

*public static void main(String[] args) {*

*Employee emp = new Employee();*

*emp.setName("Divya");*

*emp.setDepartment("HR");*

*Session session = HibernateUtil.getSessionFactory().openSession();*

*Transaction tx = null;*

*try {*

*tx = session.beginTransaction();*

*session.save(emp);*

*tx.commit();*

*System.out.println(" Employee saved successfully!");*

*} catch (Exception e) {*

*if (tx != null) tx.rollback();*

*e.printStackTrace();*

*} finally {*

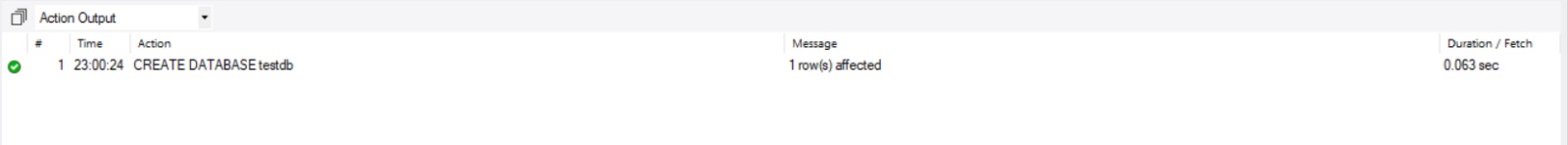
*session.close();*

*}*

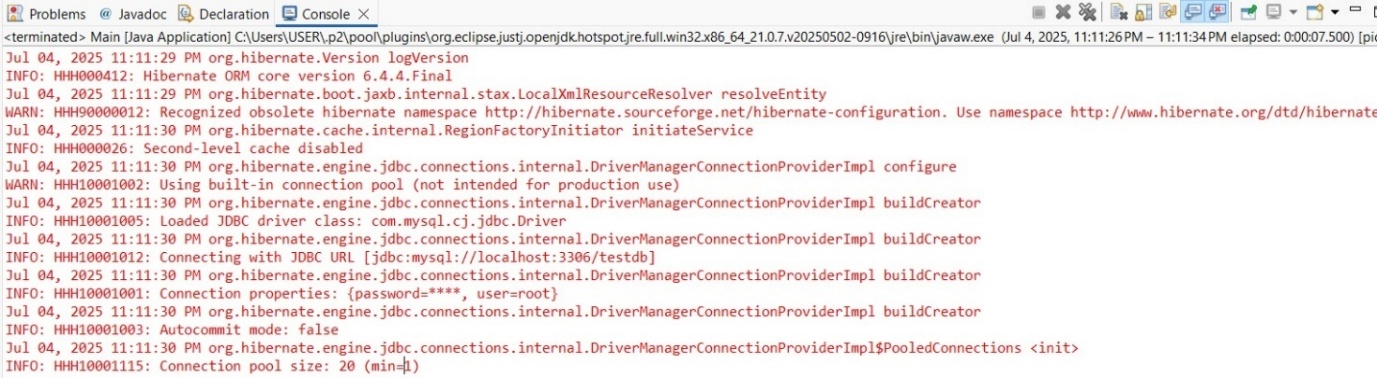
*}*

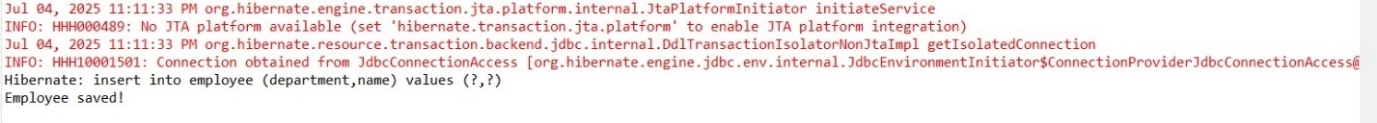
*}*

***OUTPUT:***

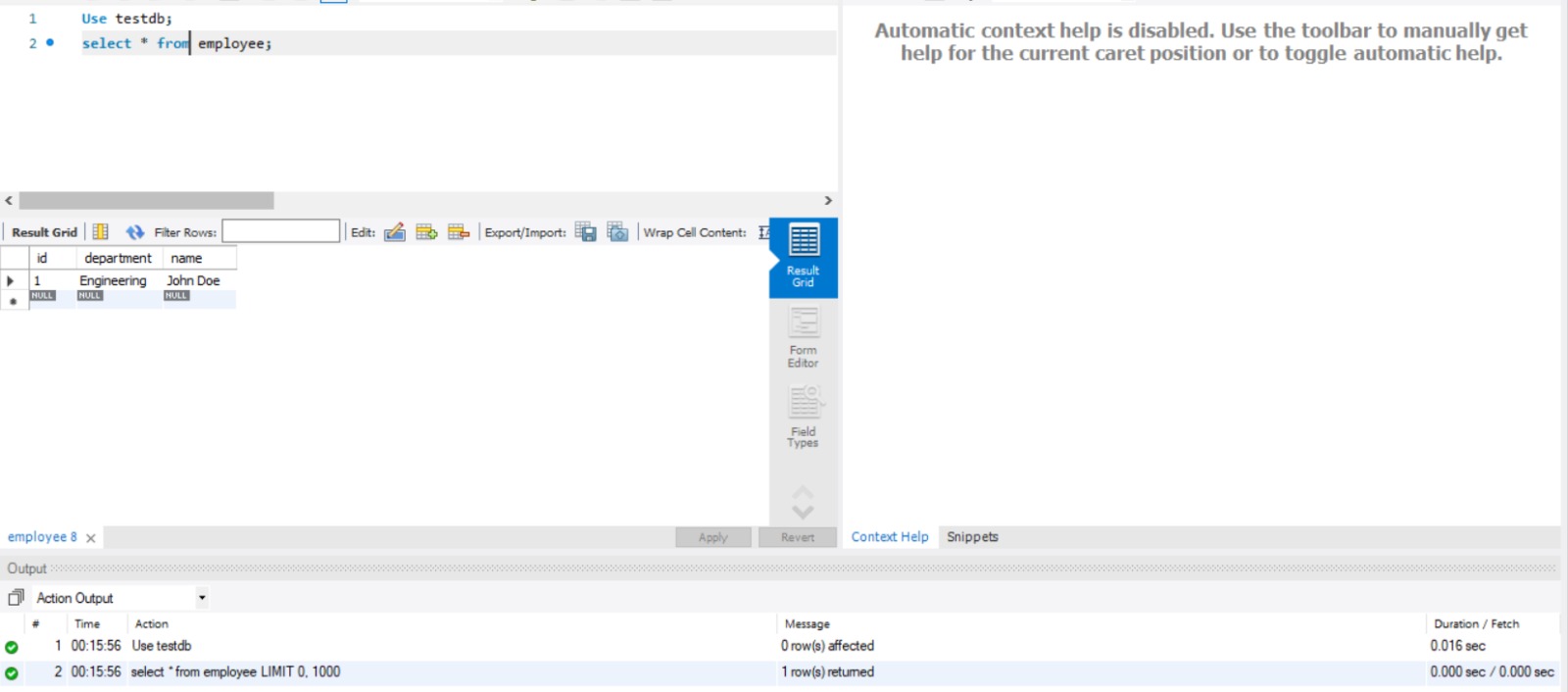


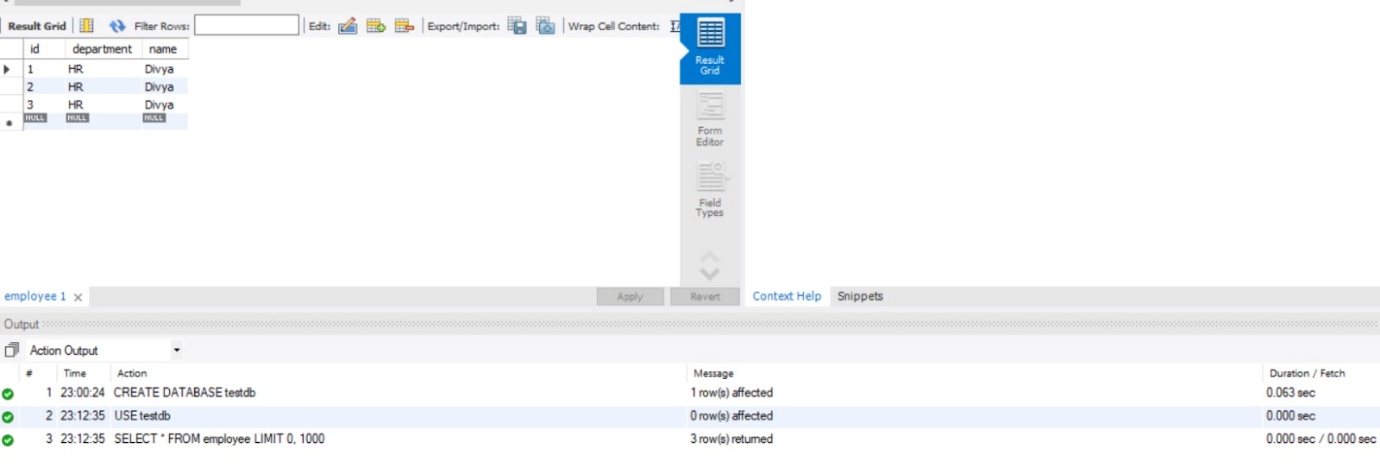
***CONSOLE OUTPUT:***





***VERIFIED IN WORKBENCH:***





**Spring Data JPA (Approach):**

***Dependencies added in pom.xml:***

*<dependencies>*

*<dependency>*

*<groupId>org.springframework.boot</groupId>*

*<artifactId>spring-boot-starter-data-jpa</artifactId>*

*</dependency>*

*<dependency>*

*<groupId>org.springframework.boot</groupId>*

*<artifactId>spring-boot-starter-web</artifactId>*

*</dependency>*

*<dependency>*

*<groupId>com.mysql</groupId>*

*<artifactId>mysql-connector-j</artifactId>*

*<version>8.0.33</version>*

*</dependency>*

*<dependency>*

*<groupId>jakarta.persistence</groupId>*

*<artifactId>jakarta.persistence-api</artifactId>*

*<version>3.1.0</version>*

*</dependency>*

*<dependency>*

*<groupId>org.springframework.boot</groupId>*

*<artifactId>spring-boot-starter-data-jpa</artifactId>*

*</dependency>*

*<dependency>*

*<groupId>org.springframework.boot</groupId>*

*<artifactId>spring-boot-starter-web</artifactId>*

*</dependency>*

*<dependency>*

*<groupId>com.mysql</groupId>*

*<artifactId>mysql-connector-j</artifactId>*

*<version>8.0.33</version>*

*</dependency>*

*</dependencies>*

***application.properties (File created in src/main/resources):***

*# Database connection*

*spring.datasource.url=jdbc:mysql://localhost:3306/testdb*

*spring.datasource.username=root*

*spring.datasource.password=\*\*\*\*\*\*\* # “ \*\*\*\*\*\*\*\*” is the password for my sql*

*spring.jpa.hibernate.ddl-auto=update*

*spring.jpa.show-sql=true*

*spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect*

*logging.level.org.hibernate.SQL=DEBUG*

***Employee.java:***

*package com.example.entity;*

*import jakarta.persistence.Entity;*

*import jakarta.persistence.Id;*

*import jakarta.persistence.GeneratedValue;*

*import jakarta.persistence.GenerationType;*

*@Entity*

*public class Employee {*

*@Id*

*@GeneratedValue(strategy = GenerationType.IDENTITY)*

*private Integer id;*

*private String name;*

*private String department;*

*public void setName(String name) {*

*this.name = name;*

*}*

*public void setDepartment(String department) {*

*this.department = department;*

*}*

*// Optional: Add getters as well*

*public Integer getId() {*

*return id;*

*}*

*public String getName() {*

*return name;*

*}*

*public String getDepartment() {*

*return department;*

*}*

*}*

***EmployeeRepository.java:***

*package com.example.repository;*

*import com.example.entity.Employee;*

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

*public interface EmployeeRepository extends JpaRepository<Employee, Integer> {*

*}*

***EmployeeService.java:***

*package com.example.service;*

*import com.example.entity.Employee;*

*import com.example.repository.EmployeeRepository;*

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

*import org.springframework.stereotype.Service;*

*@Service*

*public class EmployeeService {*

*@Autowired*

*private EmployeeRepository repository;*

*public void saveEmployee(Employee employee) {*

*repository.save(employee);*

*System.out.println("Employee saved!");*

*}*

*}*

***SpringDataJpaDemoApplication.java:***

*package com.example;*

*import com.example.entity.Employee;*

*import com.example.repository.EmployeeRepository;*

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

*import org.springframework.boot.CommandLineRunner;*

*import org.springframework.boot.SpringApplication;*

*import org.springframework.boot.autoconfigure.SpringBootApplication;*

*@SpringBootApplication*

*public class SpringDataJpaDemoApplication implements CommandLineRunner {*

*@Autowired*

*private EmployeeRepository employeeRepository;*

*public static void main(String[] args) {*

*SpringApplication.run(SpringDataJpaDemoApplication.class, args);*

*}*

*@Override*

*public void run(String... args) throws Exception {*

*Employee emp = new Employee();*

*emp.setName("John Doe");*

*emp.setDepartment("Engineering");*

*employeeRepository.save(emp);*

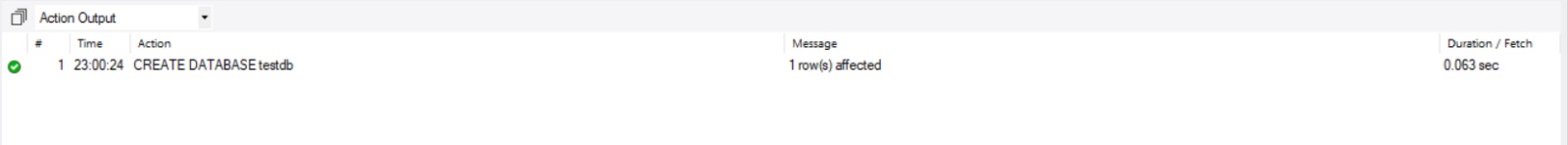
*System.out.println("Employee Saved!");*

*}*

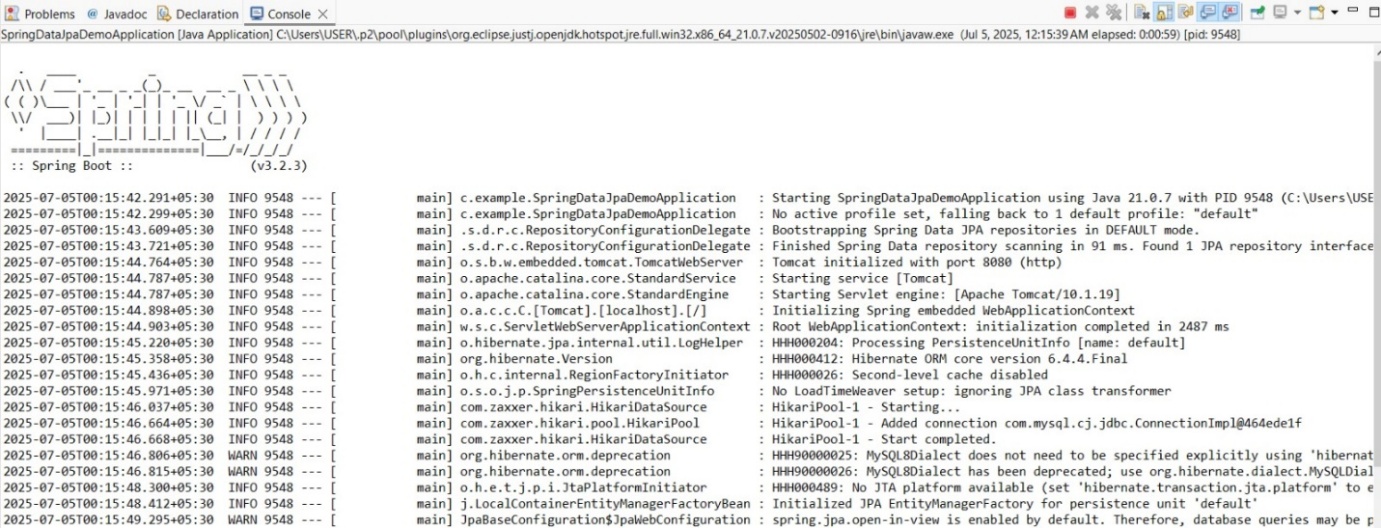
*}*

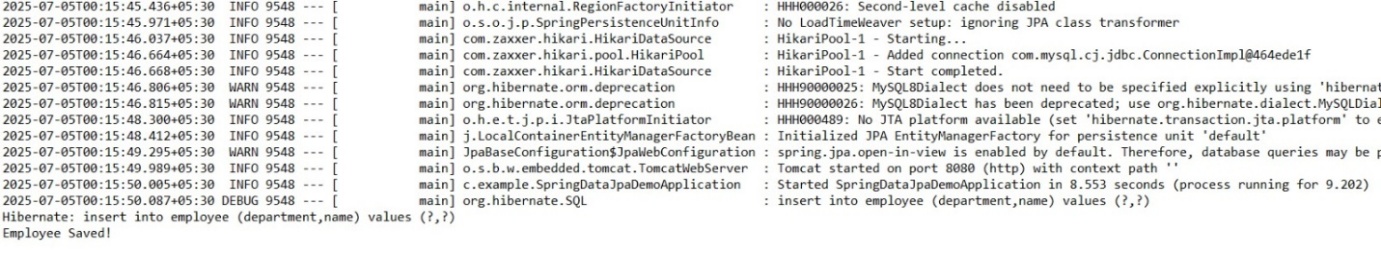
***OUTPUT:***

Created database testdb:

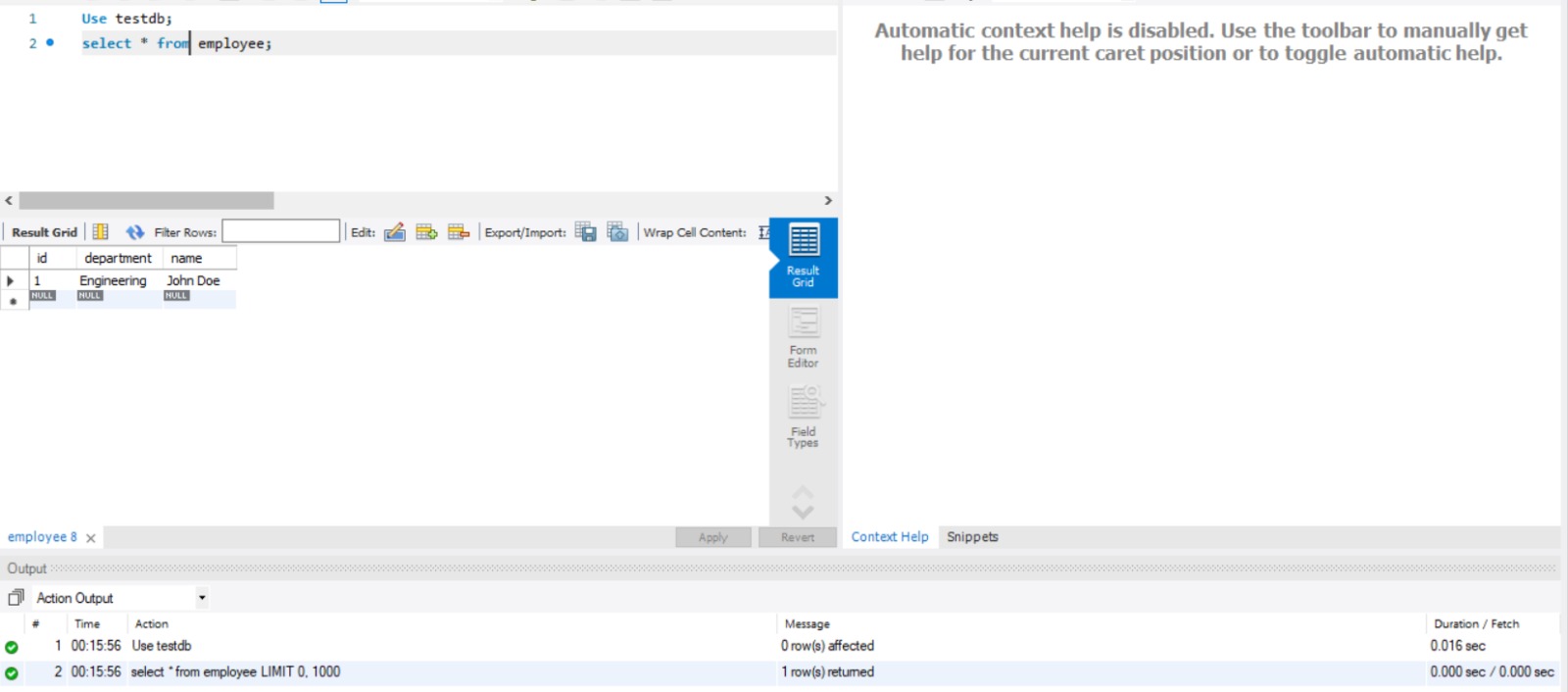


*Console output after running SpringDataJpaDemoApplication as Java Application:*





**Verified in MYSQL workbench:**



***EXPLANATION:***

***Explanation:****• Spring Data JPA simplifies data access in Java applications by providing a higher-level abstraction over JPA and Hibernate.  
• It reduces boilerplate code significantly by offering ready-made repository interfaces to handle common CRUD operations automatically.  
• In this project, a basic Spring Data JPA setup was implemented using Eclipse and Maven. The application was configured via the application.properties file.  
• Entity classes and repository interfaces were created to map and persist data.  
• Finally, successful insertion of records into the MySQL database was verified.*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*