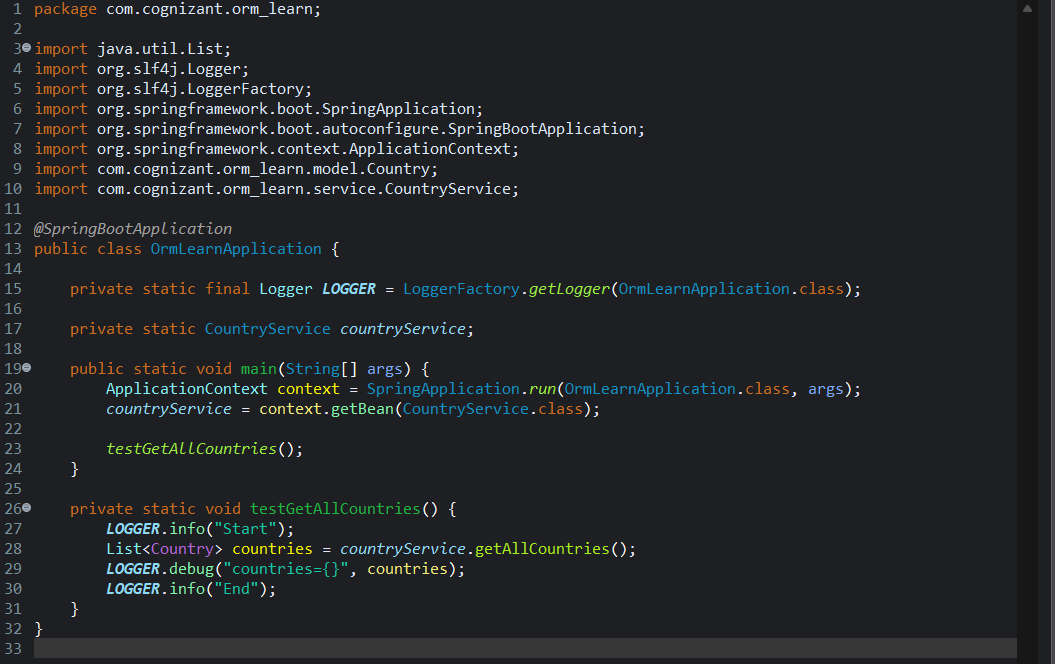
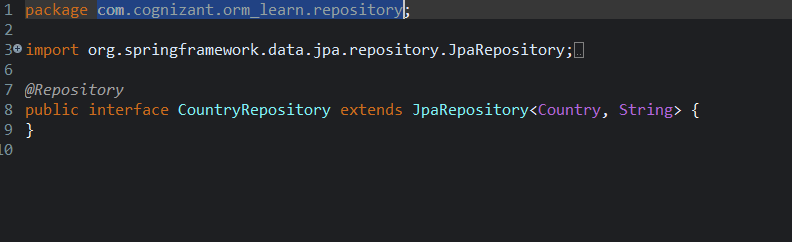
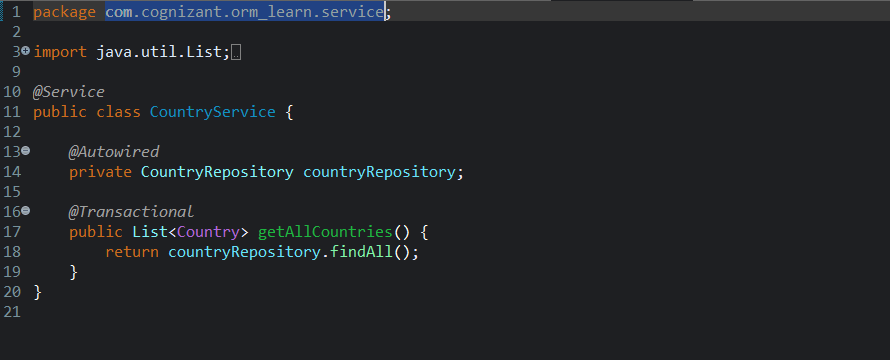
**Spring Data JPA - Quick Example**   
  
**Software Pre-requisites**

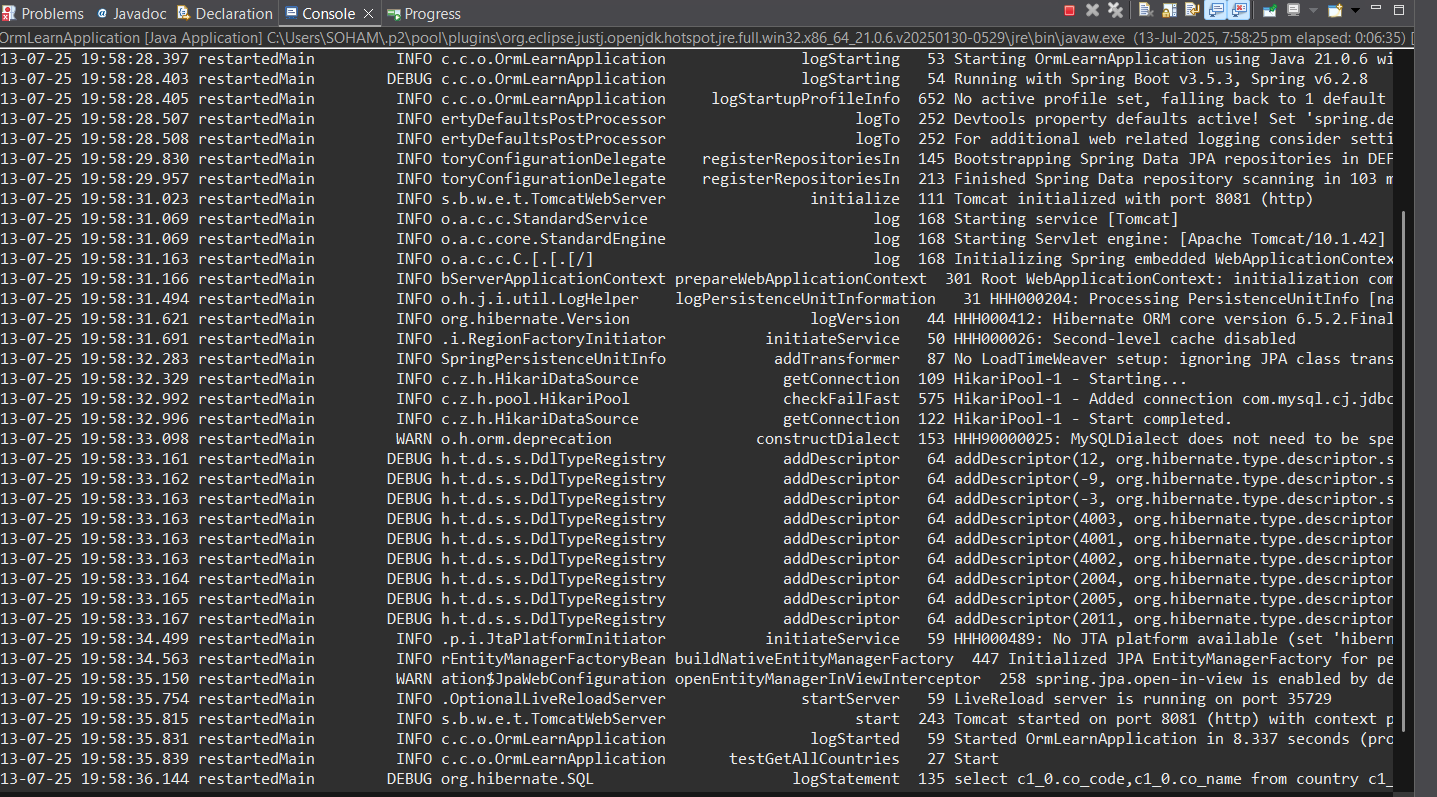
* MySQL Server 8.0
* MySQL Workbench 8
* Eclipse IDE for Enterprise Java Developers 2019-03 R
* Maven 3.6.2

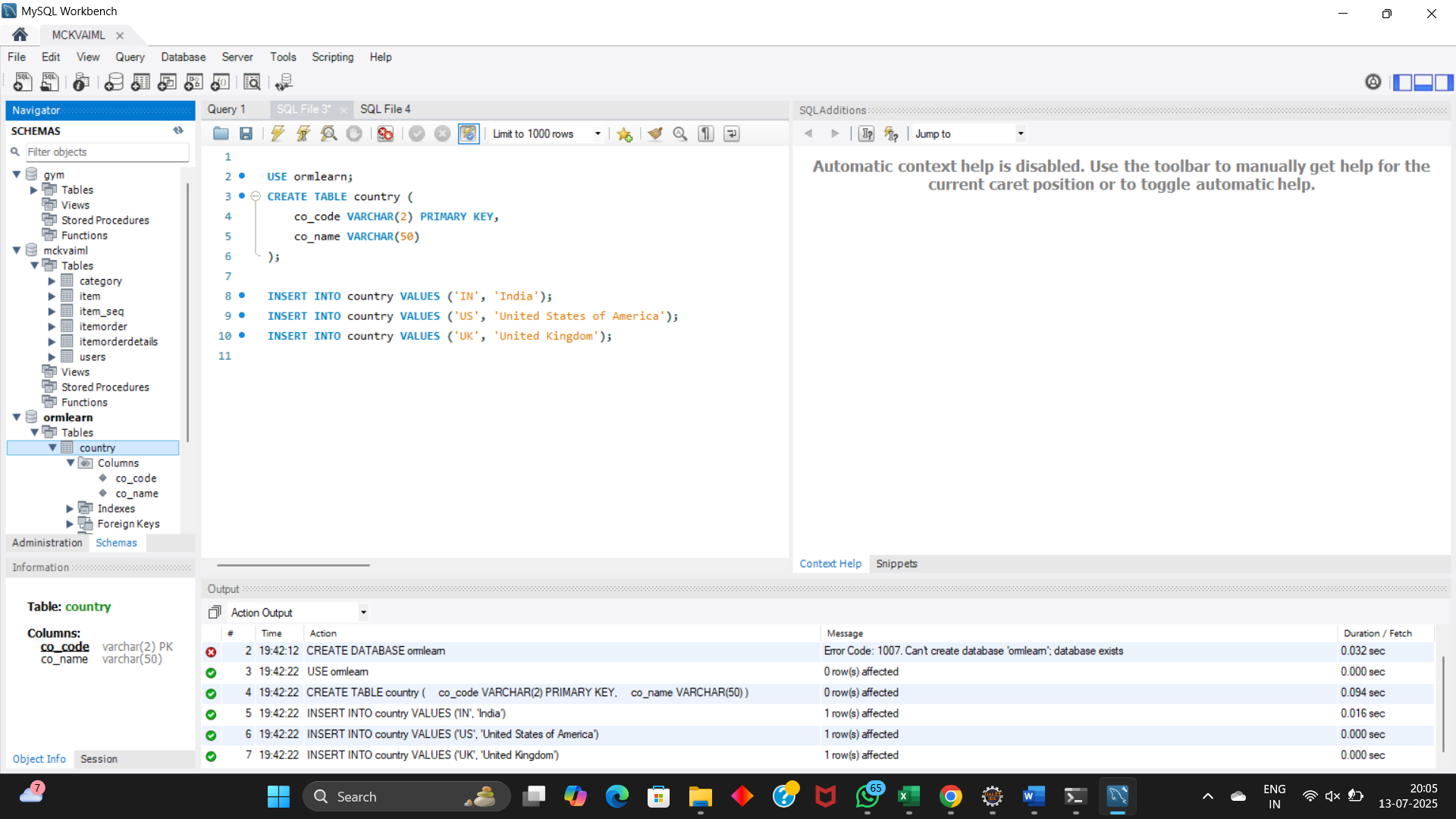












**Hands-on 4: Difference between JPA, Hibernate, and Spring Data JPA**

**Java Persistence API (JPA)**

* JPA is a **specification (JSR 338)** for persisting, reading, and managing data from Java objects.
* It **defines interfaces and annotations** but **does not provide an actual implementation.**
* Popular JPA implementations include Hibernate, EclipseLink, and OpenJPA.
* JPA helps map Java objects to relational database tables.

**Hibernate**

* Hibernate is **an Object Relational Mapping (ORM) tool** that **implements the JPA specification.**
* It provides the actual implementation for the features defined by JPA.
* Hibernate allows developers to perform CRUD operations, manage sessions, and handle transactions.
* Developers typically write boilerplate code to manage sessions and transactions.

**Spring Data JPA**

* Spring Data JPA is **a layer of abstraction over JPA and its implementation like Hibernate.**
* It **does not implement JPA itself** but **simplifies the usage of JPA/Hibernate** through repository interfaces.
* Removes the need for boilerplate code like session management, transaction handling, and exception management.
* Spring automatically manages transactions and provides built-in CRUD operations using JpaRepository.

**Example Comparison: Hibernate vs Spring Data JPA**

**Hibernate Example:**

java

CopyEdit

/\* Method to CREATE an employee in the database \*/

public Integer addEmployee(Employee employee){

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

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

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**Spring Data JPA Example:**

**EmployeeRepository.java**

java

CopyEdit

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

java

CopyEdit

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Summary of Differences:**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification | JPA Implementation | Abstraction over JPA |
| Provides actual implementation? | No | Yes | No (uses Hibernate/EclipseLink) |
| Boilerplate code | N/A | Required | Removed |
| Transaction management | Needs implementation | Manually managed | Automatic |
| CRUD operations | Interfaces only | Requires code | Auto-generated via repositories |
| Usage complexity | Medium | High | Low |