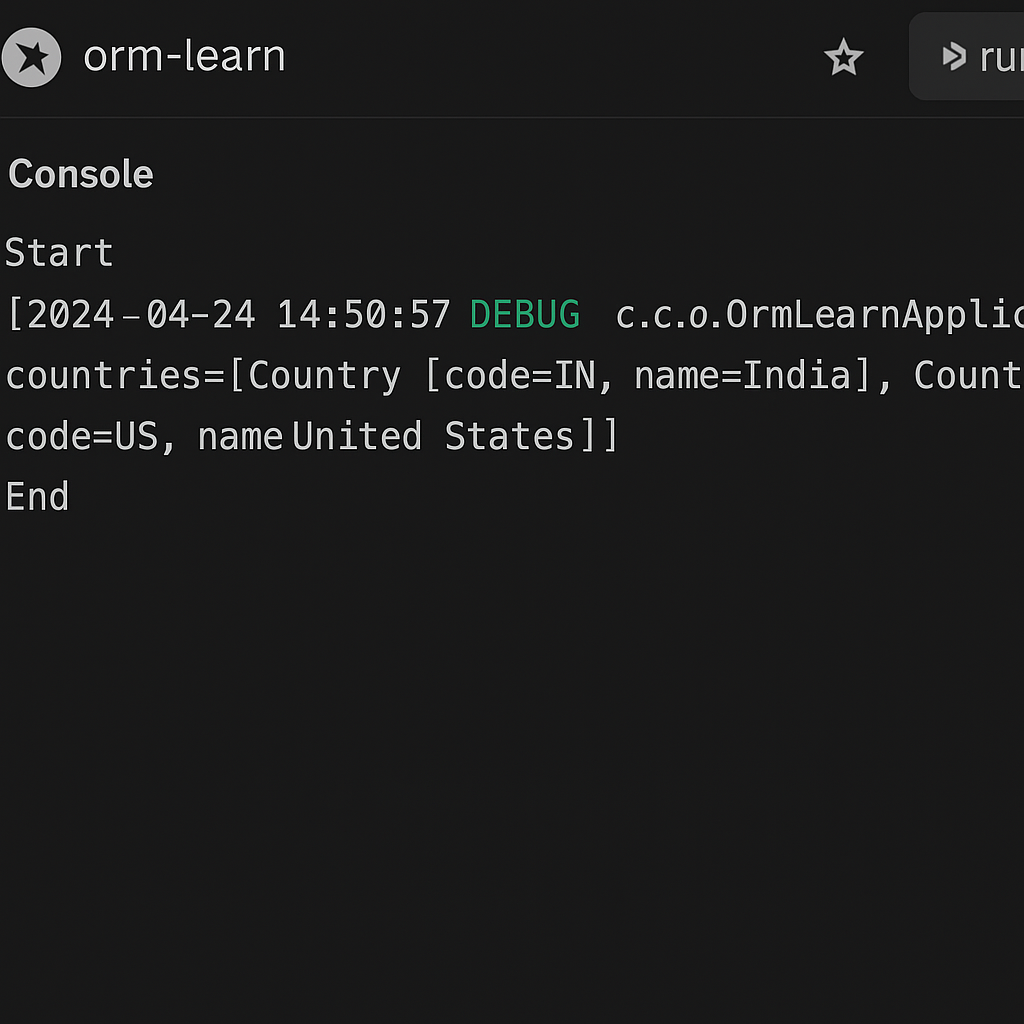
**Week 3 – Module 6**

1. **Spring Data JPA - Quick Example** :

After following all the steps mentioned in the handson, the following output has been obtained.

**OUTPUT:**



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

### **JPA (Java Persistence API)**

JPA is a specification provided by Java . It defines a standard way to map Java objects (entities) to relational database tables using annotations and APIs. JPA itself does not contain any implementation code—it is just an interface or a set of rules. It provides key annotations like @Entity, @Id, @OneToMany, and interfaces such as EntityManager, Query, and EntityTransaction. To use JPA in a project, you must use a JPA provider (like Hibernate or EclipseLink) that implements these interfaces. JPA is great for creating a database-agnostic persistence layer, but using it directly involves more boilerplate code such as manually writing repository logic and managing queries.

### **Hibernate**

Hibernate is a popular ORM framework that implements the JPA specification, and also offers many additional features beyond JPA. You can use Hibernate in two ways: either as a pure JPA provider or by using its native APIs like Session, Criteria, and Transaction. Hibernate automates the mapping between Java classes and database tables, handles SQL generation, caching, lazy loading, and more. It also supports features like first and second-level caching, batch processing, multi-tenancy, and advanced audit logging. While it follows the JPA spec, developers often use Hibernate-specific features for more flexibility and performance tuning. Hibernate can work both with or without Spring, although it’s commonly paired with Spring Boot today.

### **Spring Data JPA**

Spring Data JPA is part of the Spring Data project and acts as a higher-level abstraction over JPA. It is not an implementation of JPA, but rather a library that sits on top of a JPA provider like Hibernate. Its main purpose is to eliminate boilerplate code and make working with JPA simpler and more powerful, especially when integrated with Spring Boot. Instead of writing your own repository implementations, you can just extend interfaces like JpaRepository, CrudRepository, or PagingAndSortingRepository, and Spring Data JPA will generate the implementations for you. It also supports query derivation from method names, the use of the @Query annotation for custom JPQL/SQL, and automatic integration with Spring’s transaction management. It’s the easiest and fastest way to build a JPA-based persistence layer in modern Spring applications.