Assignment: Hands-on 4 – JPA vs Hibernate vs Spring Data JPA

1. Java Persistence API (JPA)

- JPA is a specification (JSR 338) for mapping Java objects to relational databases.
- It does not provide implementation but defines how persistence should work in Java applications.
- Requires a provider (like Hibernate, EclipseLink).

2. Hibernate

- Hibernate is an implementation of JPA and a full ORM tool.
- Manages database operations, caching, and transaction control.
- Can work with or without JPA.

3. Spring Data JPA

- Spring Data JPA is a layer over JPA (and its implementation like Hibernate).
- Reduces boilerplate code.
- Offers out-of-the-box CRUD operations via repository interfaces.

4. Code Comparison

Using Hibernate:

```
Session session = factory.openSession();

Transaction tx = session.beginTransaction();

session.save(employee);

tx.commit();

session.close();
```

Using Spring Data JPA:

```
@Autowired

private EmployeeRepository employeeRepository;
```

```
@Transactional
public void addEmployee(Employee employee) {
    employeeRepository.save(employee);
}
```

5. Conclusion Table:

Feature	**JPA**	**Hibernate**	**Spring Data JPA**
Type	Specification	Implementation	Abstraction Layer
Provides Implementation?	❌	✅	❌ (relies on Hibernate or others)
Reduces Boilerplate Code?	❌	❌	✅
Used Alone?	⚠️ Not directly	✅	✅ (with JPA provider)
Transactions	Manual or via JTA	Manual	Automatic via Spring

6. Real-World Use Cases:

Scenario	**Tool Best Suited**
Building a microservice with clean abstraction	**Spring Data JPA**
Full control over SQL and mapping	**Hibernate**
Designing ORM standard across teams	**JPA**