Q) Spring Data JPA - Quick Example

-> Spring Data JPA is a part of the Spring ecosystem that simplifies working with relational databases. It allows developers to perform CRUD operations and define custom queries using Java interfaces—without writing boilerplate SQL or JDBC code.

**Key Concepts:**

* **Entity Class**: A Java class annotated with @Entity that maps to a database table.
* **Repository Interface**: Extends JpaRepository to provide built-in CRUD methods like save(), findAll(), deleteById() etc.
* **Spring Boot Integration**: Automatically configures the database connection using application.properties.

**Benefits:**

* No need to write SQL for basic operations
* Supports pagination, sorting, custom queries
* Integrates easily with MySQL, PostgreSQL, etc.
* Reduces boilerplate DAO code

Q) Difference between JPA, Hibernate and Spring Data JPA

->

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification (Interface) | Implementation of JPA | Framework built on JPA/Hibernate |
| **Provided By** | Oracle (part of Java EE) | Red Hat | Spring team |
| **Purpose** | Defines standard ORM API | Actual ORM tool that implements JPA | Simplifies data access using repositories |
| **Usage** | Used via annotations like @Entity, @Id, etc. | You use Hibernate directly or via JPA | You use interfaces like JpaRepository |
| **Boilerplate** | Some manual setup needed | Requires more config/code | Eliminates most boilerplate |
| **Querying** | JPQL or Criteria API | HQL (Hibernate Query Language) | Method naming + JPQL queries |
| **Example** | @Entity, EntityManager | SessionFactory, HQL | CountryRepository extends JpaRepository |