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

**1. JPA – Java Persistence API**

* Who created it?
  + Defined by Oracle as part of the Java EE specification.
* What is it?
  + It's a standard/specification for ORM in Java.
  + It defines how Java objects should be mapped to database tables.
* Why is it important?
  + It allows you to write database-independent code.
  + It provides a uniform interface for persistence across different providers (like Hibernate, EclipseLink, etc.).
* How do you use it?
  + You annotate your Java classes with JPA annotations like @Entity, @Id, etc.
  + Use EntityManager to persist/fetch data.
  + But you need a provider like Hibernate to make it work.

**🔹 2. Hibernate**

* Who created it?
  + An open-source ORM tool originally developed by Gavin King.
* What is it?
  + It's a JPA implementation and a full-featured ORM framework on its own.
* Why use Hibernate?
  + It provides extra features beyond JPA, such as:
    - Caching
    - Lazy/eager loading
    - HQL (Hibernate Query Language)
* How does it work?
  + You can use Hibernate APIs directly (e.g., Session) or let it act as a JPA provider behind the scenes.

**🔹 3. Spring Data JPA**

* Who made it?
  + Part of the Spring Framework ecosystem by Pivotal (now VMware).
* What is it?
  + A wrapper/abstraction over JPA that simplifies database access in Spring Boot apps.
* Why use it?
  + To eliminate boilerplate code — just create a repository interface, and Spring handles the implementation.
* How does it work?
  + You define:

public interface UserRepository extends JpaRepository<User, Long> {}

* + Spring automatically generates the implementation.
  + It still uses JPA + a provider like Hibernate internally.