**JPA (Jakarta Persistence API)**

1. JPA is a specification (interface) for object-relational mapping (ORM) in Java.
2. It provides annotations and APIs to manage relational data using Java objects.
3. It does not provide an implementation—only the rules.
4. Requires a JPA provider like Hibernate to work.
5. Common annotations: @Entity, @Table, @Id, @GeneratedValue.

**Hibernate**

1. Hibernate is a framework and JPA provider—it implements the JPA specification.
2. It allows ORM by mapping Java classes to database tables.
3. Supports both JPA-standard APIs and its native APIs (Session, Criteria, etc.).
4. Provides advanced features like lazy loading, first/second-level caching, and batch processing.
5. Can be used with or without JPA.

**Spring Data JPA**

1. Spring Data JPA is a Springmodule that simplifies JPA-based data access.
2. It reduces boilerplate code by auto-generating implementations for repository interfaces.
3. Works on top of JPA and requires a JPA provider (like Hibernate).
4. Provides powerful abstractions like CrudRepository, JpaRepository, and custom query support with @Query.
5. Default JPA provider used is Hibernate, but you can configure others.

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

JPA (Jakarta Persistence API) is a set of rules that explains how to connect Java objects with database tables, but it doesn't have any working code by itself. Hibernate is a popular tool that follows those rules and adds extra features like caching and lazy loading. You can use Hibernate with or without JPA. Spring Data JPA is part of the Spring framework that makes working with JPA easier by reducing the amount of code you need to write. It provides ready-made interfaces like CrudRepository and JpaRepository to handle database operations easily. In short, JPA tells what to do, Hibernate does it, and Spring Data JPA makes it even simpler to use.