| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Definition** | Java Persistence API – a **specification** for persisting Java objects to relational tables. | An **implementation of JPA specification** that provides the logic to map Java objects to relational tables. | Built on **top of JPA and Spring Framework** to simplify database access. |
| **Role** | Specifies **rules and structures** for persistence. Does not implement mapping logic itself. | Implements JPA rules, provides **ORM (Object Relational Mapping)** capabilities to map POJO classes to database tables. | Provides **abstractions and repository interfaces** to simplify CRUD operations and reduce boilerplate code. |
| **Provides** | Interfaces and annotations like **@Entity, @Id, @OneToOne**, etc. | Advanced features like **caching**, lazy loading, and direct ORM functionalities. | Easy integration with Spring Boot, **automatic query methods**, custom queries using JPQL or native SQL, and simplified data access. |
| **Implementation** | **No direct implementation** – only specification. | Actual implementation that persists data to DB following JPA rules. | Uses JPA and Hibernate under the hood to provide **higher-level abstraction** and simplifies coding effort. |
| **Usage Focus** | Defines how persistence should work in general. | Provides full-fledged ORM functionalities. | Used in Spring applications to easily perform CRUD and database operations with minimal configuration. |