Concepts used for Connect Relational Database with Spring Boot

ORM :

ORM stands for **Object-Relational Mapping**.

- It's a technique to map Java objects to relational database tables.
- ORM allows developers to interact with a database using Java classes and objects instead of SQL queries.

Benefits of ORM:

- Less boilerplate code
- Database independence
- Easy CRUD operations
- Cleaner and maintainable code

Example:

```
@Entity
public class Student
{
    @Id
    private int id;
    private String name;
}
```

JPA :

JPA stands for **Java Persistence API**.

- It is a **specification** (not an implementation) provided by Java to manage relational data using Java objects.
- JPA annotations help map Java classes to database tables.

Key Annotations:

- `@Entity`: Declares the class as a JPA entity
- `@Id`: Marks primary key
- `@GeneratedValue`: Auto-increment
- `@Column`: Customize column name

Hibernate vs JPA

Feature	JPA	Hibernate
Type	Specification	Implementation of JPA
Provided by	Java EE	Red Hat
Usage	Used with Hibernate, EclipseLink	Used with JPA or native
Annotation	Uses JPA standard annotations	Uses JPA + Hibernate-specific

Spring Data JPA

- **Spring Data JPA** is part of Spring Data project.
- It simplifies JPA implementation by reducing boilerplate code.
- Auto-generates common queries using method names.

Key Interface: `JpaRepository<T, ID>`

Example:

```
public interface StudentRepository extends JpaRepository<Student,
Integer>
{
    List<Student> findByName(String name);
}
```

Spring Boot auto-configures JPA with minimal setup.

Configuration for Spring Data JPA with MySQL

properties

```
# application.properties
spring.datasource.url=jdbc:mysql://localhost:3306/marvellous
spring.datasource.username=root
spring.datasource.password=your_password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5D
ialect
```