POC (Proof of Concept)

Creating a Proof of Concept (POC) for the relationships: @OneToOne, @OneToMany, @ManyToOne, and @ManyToMany. We will define four entities: Author, Address, Book, and Category, and set up the relationships between them.

Step 1: Create the Maven Project

Ensure your pom.xml has the necessary dependencies:

> Spring-JPA

> Spring-Web

> Postgresql

Step 2: Configure PostgreSQL in application.properties

In src/main/resources/application.properties, add your database configuration:

- spring.datasource.url=jdbc:postgresql://localhost:5432/Books_Management
- spring.datasource.username=postgres
- spring.datasource.password=Akki@123
- spring.datasource.driver-class-name=org.postgresql.Driver

- spring.jpa.hibernate.ddl-auto=update
- spring.jpa.show-sql=true
- spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

Step 3: Create Entity Classes

1. @OneToOne Relationship

   An Author has one Address.

   An Address belongs to one Author.

Author Entity

```
@Entity
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
    private String bio;

    @OneToOne(mappedBy = "author", cascade = CascadeType.ALL)
    private Address address;

}
```

Address Entity

```
@Entity
public class Address {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String city;
    private String state;
    private String zipcode;

    @OneToOne
    @JoinColumn(name = "author_id")
    private Author author;
}
```

2. @OneToMany and @ManyToOne Relationship

An Author can have many Books.

A Book belongs to one Author.

Book Entity

```
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String publishedDate;

    @ManyToOne
    @JoinColumn(name = "author_id")
    private Author author;
}
```

3. @ManyToMany Relationship

A Book can belong to many Categories.

A Category can have many Books.

Category Entity

```
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String description;
    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "category")
    private List<Book> books;
}
```