### 1. Creating JPA Entities

#### 1.1 Define the Employee Entity

The Employee entity should have fields for id, name, email, and a relationship to the Department.

**Employee.java**

package com.example.employeemanagementsystem.model;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.\*;

@Entity

@Table(name = "employees")

@Data

@NoArgsConstructor

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id", nullable = false)

private Department department;

// Add constructors, getters, and setters if not using Lombok

}

* @Entity: Specifies that this class is a JPA entity.
* @Table(name = "employees"): Maps the entity to the "employees" table.
* @Id: Indicates the primary key of the entity.
* @GeneratedValue(strategy = GenerationType.IDENTITY): Specifies the strategy for generating primary key values.
* @ManyToOne: Defines a many-to-one relationship with the Department entity.
* @JoinColumn(name = "department\_id"): Specifies the column in the employees table that holds the foreign key to the Department entity.

#### 1.2 Define the Department Entity

The Department entity should have fields for id and name. It will also have a one-to-many relationship with the Employee entity.

**Department.java**

package com.example.employeemanagementsystem.model;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "departments")

@Data

@NoArgsConstructor

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department")

private Set<Employee> employees;

// Add constructors, getters, and setters if not using Lombok

}

* @Entity: Specifies that this class is a JPA entity.
* @Table(name = "departments"): Maps the entity to the "departments" table.
* @Id: Indicates the primary key of the entity.
* @GeneratedValue(strategy = GenerationType.IDENTITY): Specifies the strategy for generating primary key values.
* @OneToMany(mappedBy = "department"): Defines a one-to-many relationship with the Employee entity. The mappedBy attribute specifies that the department field in the Employee entity owns the relationship.

### 2. Mapping Entities to Database Tables

The entities are already mapped to database tables using JPA annotations. Here's a brief explanation of how the mapping works:

Employee **Entity**:

* + Mapped to the employees table.
  + Has a foreign key column department\_id that references the departments table.

Department **Entity**:

* + Mapped to the departments table.
  + Has a one-to-many relationship with the Employee entity, where one department can have multiple employees.

### Example Table Structures

Given the entities above, here are the corresponding SQL table structures:

employees **Table**

| **Column** | **Type** | **Constraints** |
| --- | --- | --- |
| id | BIGINT | PRIMARY KEY, AUTO\_INCREMENT |
| name | VARCHAR(255) | NOT NULL |
| email | VARCHAR(255) | NOT NULL |
| department\_id | BIGINT | FOREIGN KEY REFERENCES departments(id) |

departments **Table**

| **Column** | **Type** | **Constraints** |
| --- | --- | --- |
| id | BIGINT | PRIMARY KEY, AUTO\_INCREMENT |
| name | VARCHAR(255) | NOT NULL |