**1. Explain the relationship between the "Product" and "Product\_Category" entities from the above diagram.**

* The "Product" and "Product\_Category" entities are likely related through a foreign key. The "category id" in the "Product" table is likely a foreign key referencing the "id" in the "Product\_Category" table. This establishes a connection between products and their respective categories.
* The relationship between the "Product" and "Product\_Category" entities is a one-to-many relationship. This means that each product can belong to only one category, but a category can have multiple products associated with it.

Example :

Entity Class Name:ProductCategory

import javax.persistence.\*;

import java.util.List;

@Entity

@Table(name = "product\_category")

public class ProductCategory {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String desc;

private LocalDate created\_at;

private LocalDate modified\_at;

private LocalDate deleted\_at;

@OneToMany(mappedBy = "category", cascade = CascadeType.ALL)

private List<Product> products=new ArrayList<Product>();

// Getters and setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public LocalDate getCreated\_at() {

return created\_at;

}

public void setCreated\_at(LocalDate created\_at) {

this. Created\_at = created\_at;

}

public LocalDate getModified\_at() {

return modified\_at;

}

public void setModified\_at(LocalDate modified\_at) {

this.modified\_at = modified\_at;

}

public LocalDate getDeleted\_at() {

return deleted\_at;

}

public void setDeleted\_at(LocalDate deleted\_at) {

this.deleted\_at = deleted\_at;

}

}

Entity Class Name:Product

import javax.persistence.\*;

@Entity

@Table(name = "product")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String desc;

private String SKU;

private double price;

private LocalDate created\_at;

private LocalDate modified\_at;

private LocalDate deleted\_at;

@ManyToOne

@JoinColumn(name = "category\_id")

private ProductCategory category;

// Getters and setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

public String getSKU() {

return SKU;

}

public void setSKU(String SKU) {

this.SKU = SKU;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public ProductCategory getCategory() {

return category;

}

public void setCategory(ProductCategory category) {

this.category = category;

}

public LocalDate getCreated\_at() {

return created\_at;

}

public void setCreated\_at(LocalDate created\_at) {

this.created\_at = created\_at;

}

public LocalDate getModified\_at() {

return modified\_at;

}

public void setModified\_at(LocalDate modified\_at) {

this.modified\_at = modified\_at;

}

public LocalDate getDeleted\_at() {

return deleted\_at;

}

public void setDeleted\_at(LocalDate deleted\_at) {

this.deleted\_at = deleted\_at;

}}

**2. How could you ensure that each product in the "Product" table has a valid category assigned to it?**

**Ex:**

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **Category\_id** |
| 1 | mobile | 1 |
| 2 | laptop | 1 |
| 3 | charger | 2 |

* To ensure that each product in the "Product" table has a valid category assigned to it, you can use a foreign key constraint. This constraint will enforce that the value in the "category\_id" column of the "Product" table must match a valid category ID from the "Product\_Category" table.

**3. Create schema in any Database script or any ORM (Object Relational Mapping).**

**ORM:** Object Relational Mapping

ORM is used to perform CRUD Operation in data base.

1. Create Product\_Category table

CREATE TABLE Product\_Category (

id INT PRIMARY KEY,

name VARCHAR(255),

desc TEXT,

created\_at TIMESTAMP,

modified\_at TIMESTAMP,

deleted\_at TIMESTAMP

);

2.Create Product table

CREATE TABLE Product (

id INT PRIMARY KEY,

name VARCHAR(255),

desc TEXT,

created\_at TIMESTAMP,

modified\_at TIMESTAMP,

deleted\_at TIMESTAMP,

category\_id INT,

FOREIGN KEY (category\_id) REFERENCES Product\_Category(id)

);

3. Create Product\_Inventory table

CREATE TABLE Product\_Inventory (

id INT PRIMARY KEY,

SKU VARCHAR(255),

quantity INT,

category\_id INT,

created\_at TIMESTAMP,

modified\_at TIMESTAMP,

price DECIMAL,

deleted\_at TIMESTAMP,

FOREIGN KEY (category\_id) REFERENCES Product\_Category(id)

);

4.Create Discount table

CREATE TABLE Discount (

id INT PRIMARY KEY,

name VARCHAR(255),

desc TEXT,

discount\_percent DECIMAL,

active BOOLEAN,

created\_at TIMESTAMP,

modified\_at TIMESTAMP,

deleted\_at TIMESTAMP

);

5. Link Product and Discount tables

CREATE TABLE Product\_Discount (

product\_id INT,

discount\_id INT,

PRIMARY KEY (product\_id, discount\_id),

FOREIGN KEY (product\_id) REFERENCES Product(id),

FOREIGN KEY (discount\_id) REFERENCES Discount(id)

);