# E-Commerce System - Database Schema & Relations

## User

- id (PK)
- name
- email
- role (CUSTOMER, SELLER, ADMIN)
- password
- phone
- createdAt

## Category

- id (PK)
- name
- description

## Product

- id (PK)
- name
- description
- price
- stockQuantity
- seller_id (FK → User)
- category_id (FK → Category)

## Review

- id (PK)
- rating (1–5)
- comment
- user_id (FK → User)
- product_id (FK → Product)
- createdAt

## Cart

- id (PK)
- user_id (FK → User, unique)

## CartItem

- id (PK)
- cart_id (FK → Cart)
- product_id (FK → Product)

- quantity

## Order

- id (PK)
- orderNumber
- status (PLACED, CONFIRMED, SHIPPED, DELIVERED, CANCELLED)
- user_id (FK → User)
- createdAt
- updatedAt

## OrderItem

- id (PK)
- order_id (FK → Order)
- product_id (FK → Product)
- quantity
- priceAtPurchase

## Payment

- id (PK)
- amount
- method (CREDIT_CARD, UPI, NET_BANKING, COD)
- status (PENDING, SUCCESS, FAILED, REFUNDED)
- order_id (FK → Order)
- createdAt

## Shipment

- id (PK)
- trackingNumber
- carrier (FedEx, DHL, etc.)
- status (PENDING, IN_TRANSIT, DELIVERED)
- order_id (FK → Order)

## Relations Recap

- User (Customer) → Cart (1:1)
- User (Customer) → Order (1:M)
- User (Seller) → Product (1:M)
- Category → Product (1:M)
- Product → Review (1:M)
- Cart → CartItem (1:M)
- Order → OrderItem (1:M)
- Order → Payment (1:1)
- Order → Shipment (1:1)

## Features to Implement

- Shopping Cart (add/remove/update items, calculate total)

- Order Placement (convert cart to order, update stock)
- Product Reviews (customer ratings & comments)
- Order Tracking (check status, shipment details)
- Admin Analytics (top-selling products, revenue reports)

## Complex Queries to Practice

**Top 5 best-selling products**

```
SELECT oi.product.id, SUM(oi.quantity) FROM OrderItem oi GROUP BY
oi.product.id ORDER BY SUM(oi.quantity) DESC
```

**Get all pending shipments for a customer**

```
SELECT s FROM Shipment s JOIN s.order o WHERE o.user.id = :customerId AND
s.status != 'DELIVERED'
```

**Find customers who spent the most in last 30 days**

```
SELECT o.user.id, SUM(p.amount) FROM Payment p JOIN p.order o WHERE
p.status = 'SUCCESS' AND p.createdAt >= CURRENT_DATE - 30 GROUP BY
o.user.id ORDER BY SUM(p.amount) DESC
```

**Get products with average rating > 4**

```
SELECT r.product.id, AVG(r.rating) FROM Review r GROUP BY r.product.id
HAVING AVG(r.rating) > 4
```

**Find orders with at least 3 different products**

```
SELECT o.id FROM OrderItem oi JOIN oi.order o GROUP BY o.id HAVING
COUNT(DISTINCT oi.product.id) >= 3
```