|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2차시 |  |  |  | 실습: □ | 과제 :  | 평가 : □ |
| 과제)  쇼핑몰을 운영하는 데 필요한 기본적인 관계형 데이터베이스 스키마를 설계할 때, 여러 핵심 테이블을 아래 내용을 참고하여 테이블 스키마를 설계하여 보세요.  필드는 자유롭게 구성하세요.  1. Customers (고객 정보)  customer\_id: 고객의 고유 식별자 (기본 키)  2. Products (제품 정보)  product\_id: 제품의 고유 식별자 (기본 키)  3. Categories (카테고리 정보)  category\_id: 카테고리의 고유 식별자 (기본 키)  4. Orders (주문 정보)  order\_id: 주문의 고유 식별자 (기본 키)  5. OrderDetails (주문 상세 정보)  order\_detail\_id: 주문 상세 정보의 고유 식별자 (기본 키)  관계구조:   * Customers와 Orders: 하나의 고객은 여러 주문을 할 수 있습니다 (1관계). * Orders와 OrderDetails: 하나의 주문은 여러 개의 주문 상세 정보를 가질 수 있습니다 (1관계). * Products와 OrderDetails: 하나의 제품은 여러 주문 상세 정보에 포함될 수 있습니다 (1관계). * Categories와 Products: 하나의 카테고리는 여러 제품을 포함할 수 있습니다 (1관계).   -- 과제 내:  DROP DATABASE IF EXISTS shop;  CREATE DATABASE shop;  use shop;  CREATE TABLE Customers  (  customer\_id INT AUTO\_INCREMENT PRIMARY KEY,  customer\_name VARCHAR(16) NOT NULL  );  CREATE TABLE Categories  (  category\_id INT AUTO\_INCREMENT PRIMARY KEY,  category\_name VARCHAR(16) NOT NULL,  category\_name\_kor VARCHAR(16)  );  CREATE TABLE Products  (  product\_id INT AUTO\_INCREMENT PRIMARY KEY,  product\_name VARCHAR(16) NOT NULL,  category\_id INT,  price INT,  FOREIGN KEY (category\_id) REFERENCES Categories(category\_id)  );  CREATE TABLE Orders  (  order\_id INT AUTO\_INCREMENT PRIMARY KEY,  customer\_id INT NOT NULL,  quantity INT DEFAULT 1,  order\_date DATE,  status VARCHAR(16) DEFAULT '접수 중입니다',  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)  );  CREATE TABLE OrderDetails (  order\_detail\_id INT AUTO\_INCREMENT PRIMARY KEY,  order\_id INT,  product\_id INT,  quantity INT DEFAULT 1,  price INT NOT NULL,  total\_price INT AS (quantity \* price) STORED,  FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),  FOREIGN KEY (product\_id) REFERENCES Products(product\_id)  );  -- 예시 데이터:  INSERT INTO Customers (customer\_name) VALUES ('Theo');  INSERT INTO Customers (customer\_name) VALUES ('Gildong');  INSERT INTO Customers (customer\_name) VALUES ('Lilpa');  INSERT INTO Categories (category\_name, category\_name\_kor) VALUES ('Adventure', '어드벤처');  INSERT INTO Categories (category\_name, category\_name\_kor) VALUES ('RTS', '전략 시뮬레이션');  INSERT INTO Categories (category\_name, category\_name\_kor) VALUES ('Action', '액션');  INSERT INTO Products (product\_name, category\_id, price) VALUES ('Baldur\'s Gate 3', 1, 66000);  INSERT INTO Products (product\_name, category\_id, price) VALUES ('Civilization® VI', 2, 65000);  INSERT INTO Products (product\_name, category\_id, price) VALUES ('Elden Ring', 3, 89800);  INSERT INTO Orders (customer\_id, quantity, order\_date) VALUES (1, 1, '2024-07-01');  INSERT INTO Orders (customer\_id, quantity, order\_date, status) VALUES (2, 1, '2024-04-05', '주문 완료');  INSERT INTO Orders (customer\_id, quantity, order\_date, status) VALUES (3, 2, '2023-11-23', '주문 완료');  INSERT INTO OrderDetails (order\_id, product\_id, quantity, price) VALUES (1, 1, 1, 66000);  INSERT INTO OrderDetails (order\_id, product\_id, quantity, price) VALUES (2, 2, 1, 65000);  INSERT INTO OrderDetails (order\_id, product\_id, quantity, price) VALUES (3, 3, 2, 89800);  -- 예시 출력:  SELECT  o.order\_id,  c.customer\_name,  p.product\_name,  cat.category\_name,  cat.category\_name\_kor,  od.quantity,  od.price,  od.total\_price,  o.order\_date,  o.status  FROM  Orders o  JOIN  Customers c ON o.customer\_id = c.customer\_id  JOIN  OrderDetails od ON o.order\_id = od.order\_id  JOIN  Products p ON od.product\_id = p.product\_id  JOIN  Categories cat ON p.category\_id = cat.category\_id; | | | | | | |