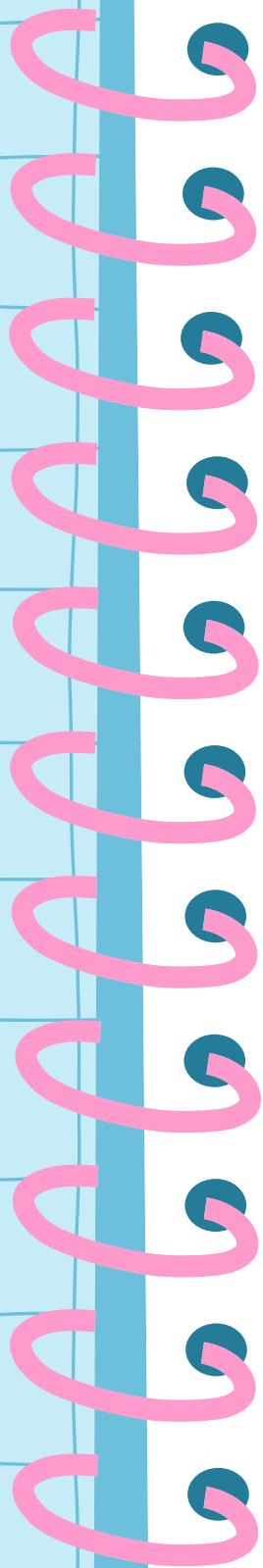


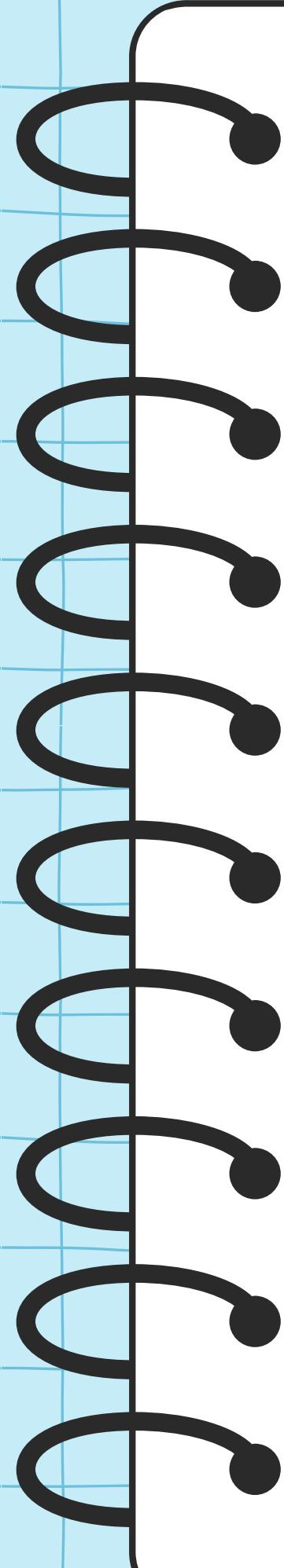
# 온라인 도서·문구 판매 관리 시스템

2팀 손한이 윤귀미 전현식 정인교 피동기

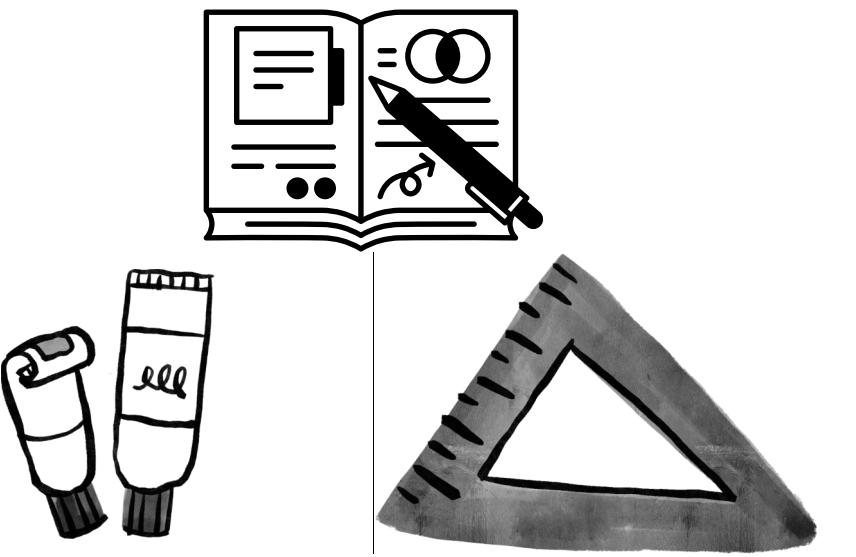


# 목차

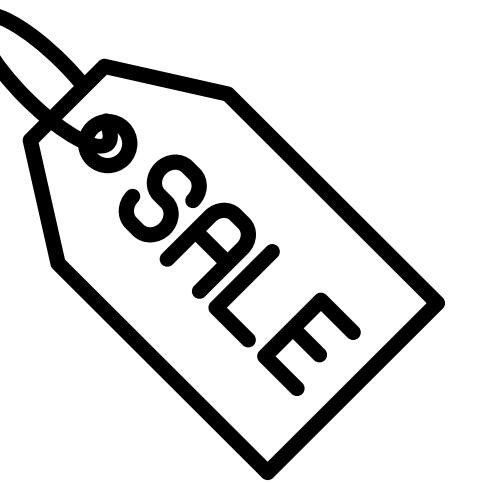
- 
- 프로젝트 개요**
  - 팀 구성원 소개 및 역할**
  - 프로젝트 수행 절차 및 결과**
  - 자체 평가 의견**



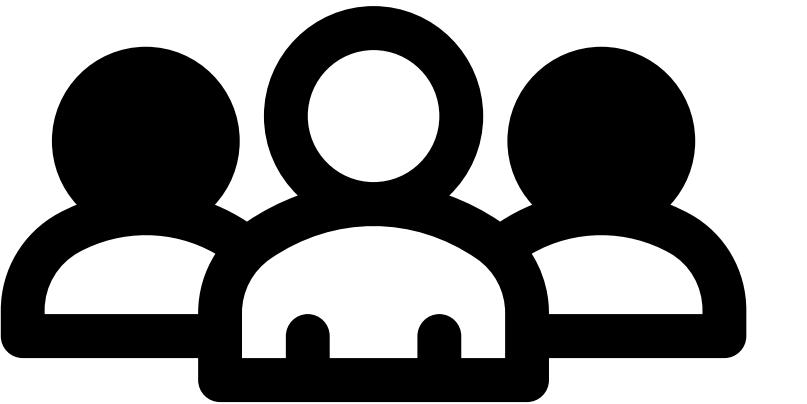
개요



도서 · 문구 제품 관리



상품 판매 관리

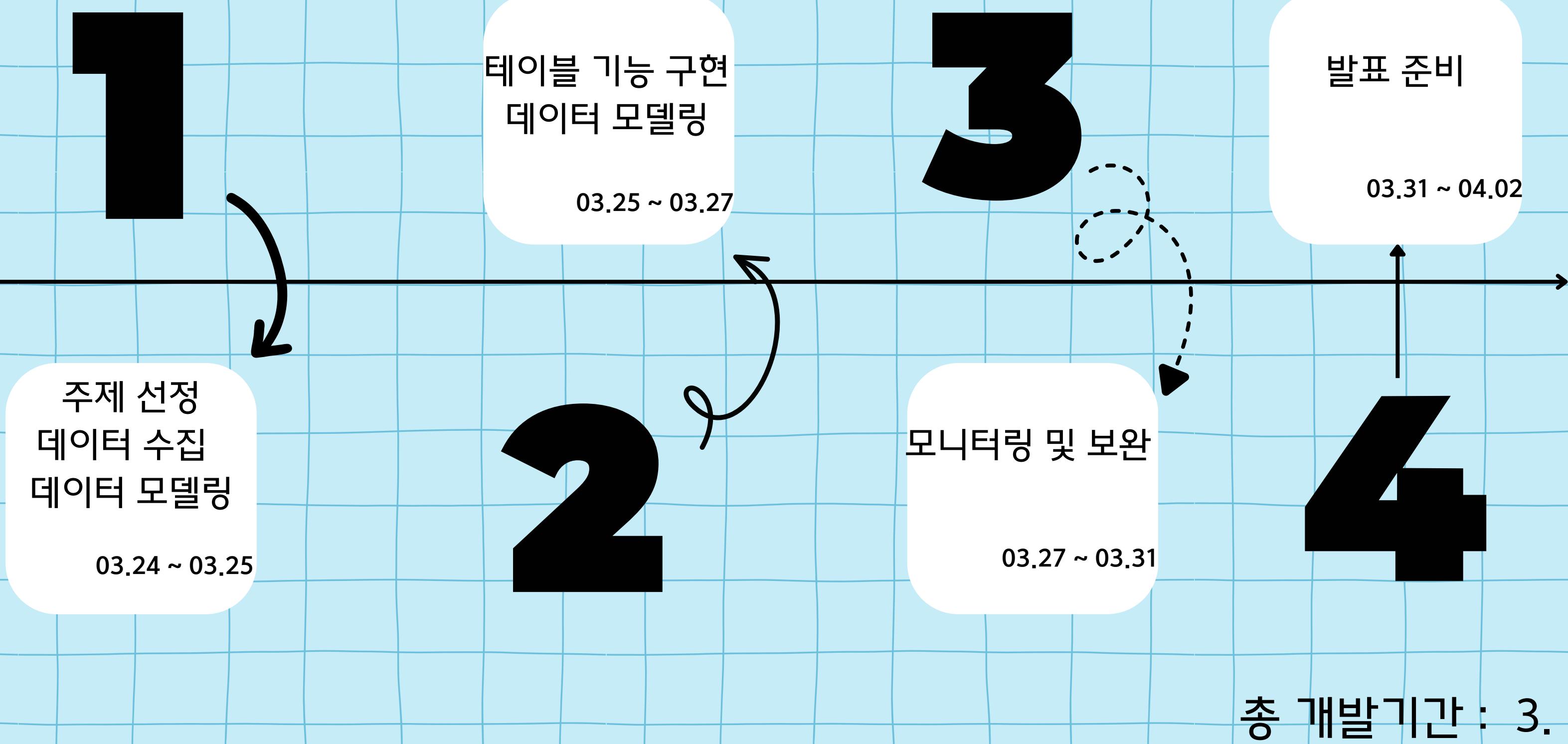


회원 관리



리뷰 및 댓글 관리

# 세부 일정



# 팀원 역할 소개

주제 선정,  
테이블 생성 및 데이터 추가



손한이

전반적인 진행 상황 점검  
및 동기화  
문구 테이블 관리, 평점  
관리 트리거 구현



윤귀미

고객 테이블 담당  
고객 이름, 도서 검색,  
등급 검색 기능  
구현



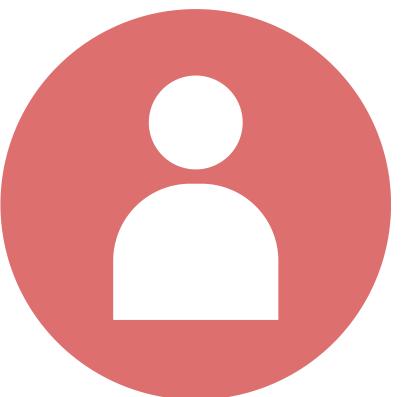
전현식

상품(책) 테이블 담당  
프로시저 및 트리거 등의  
함수를 활용한  
도서정보 검색기능 구현



정인교

상품, 주문 테이블  
회원 등급에 따른 할인 기능  
테이블, 재고 관리 트리거  
매출 검색 구현

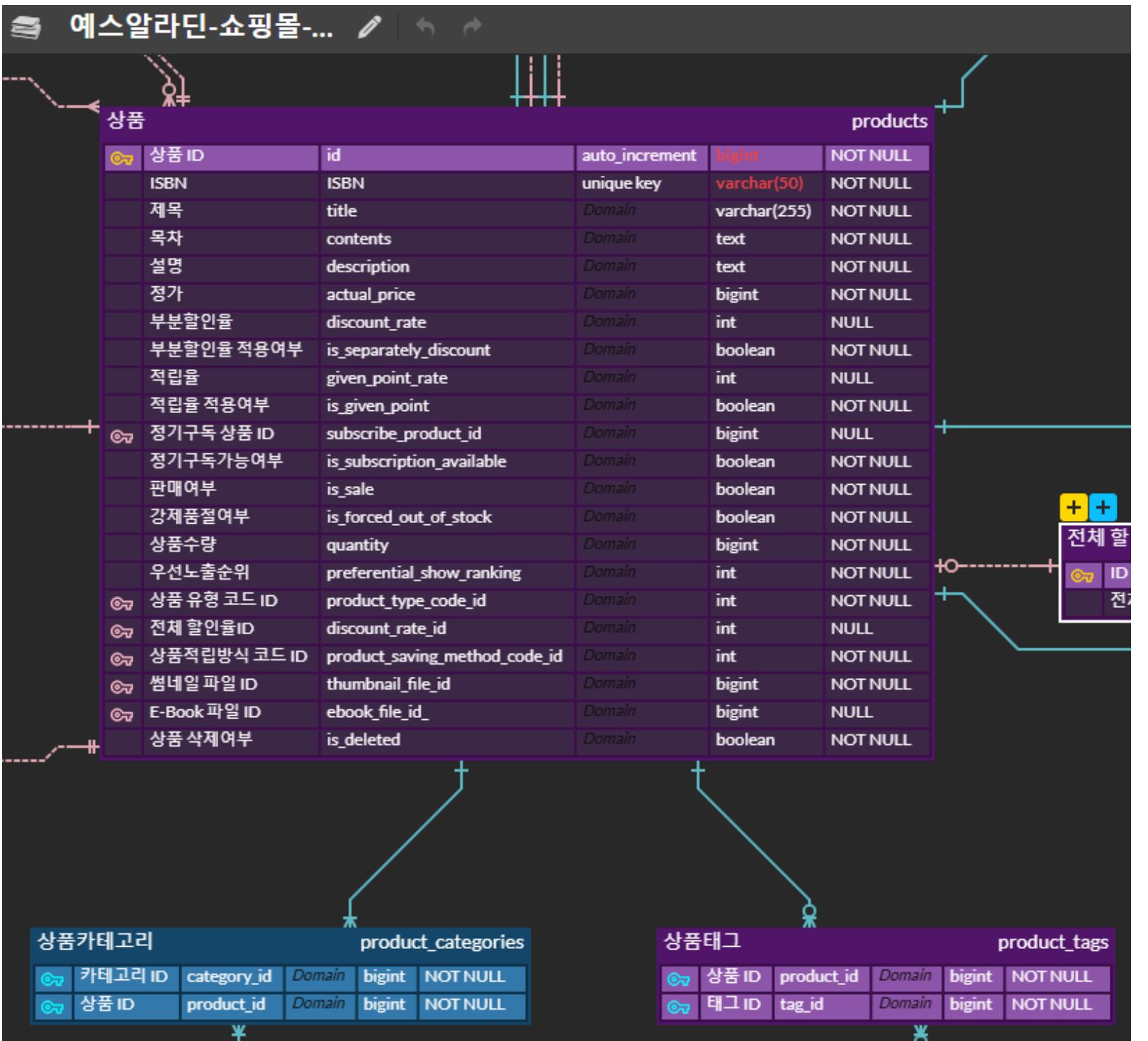


피동기

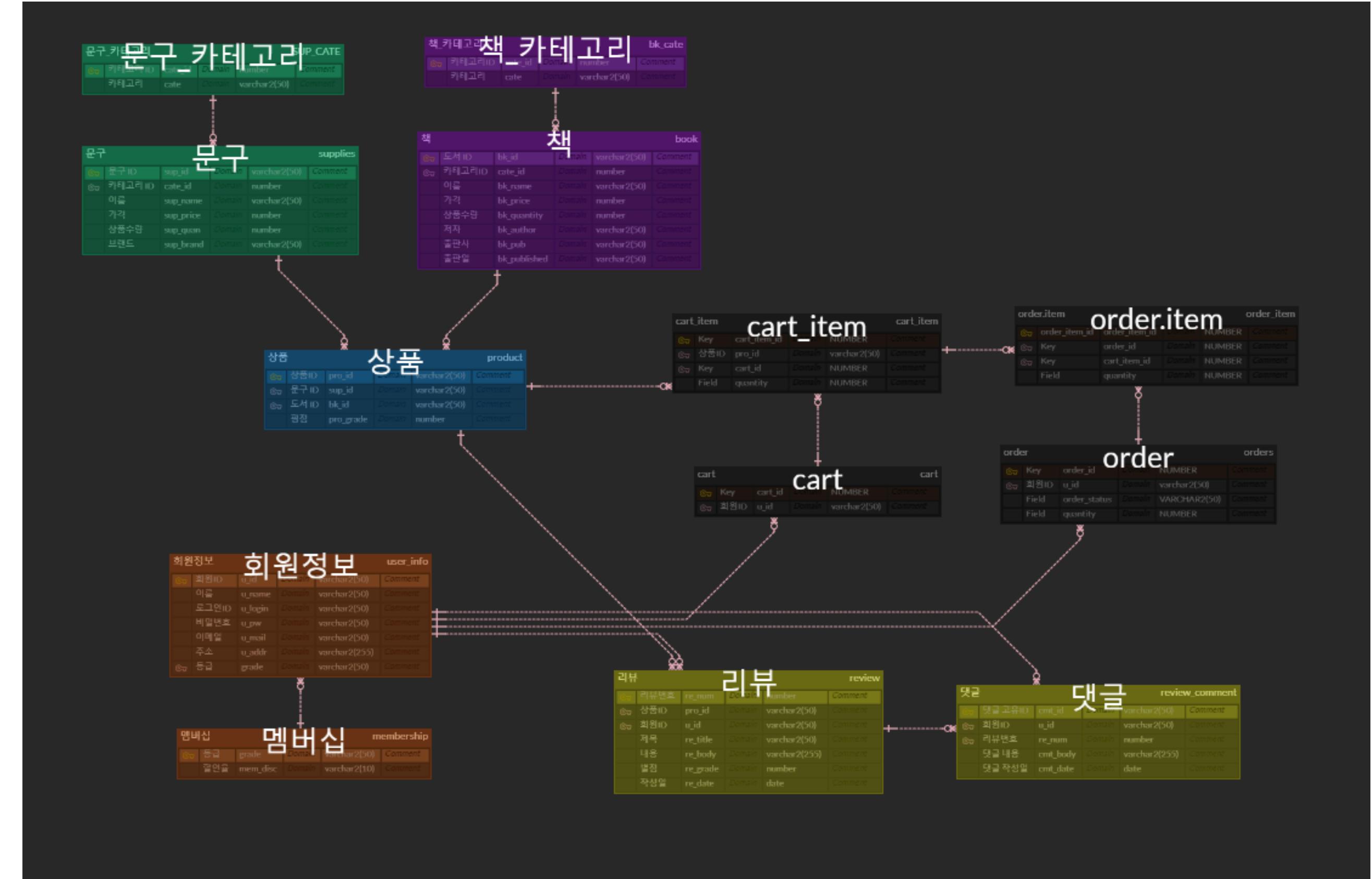
리뷰, 댓글 담당  
프로시저, 함수를 활용한  
도서검색 기능 구현

# 데이터 참고 자료

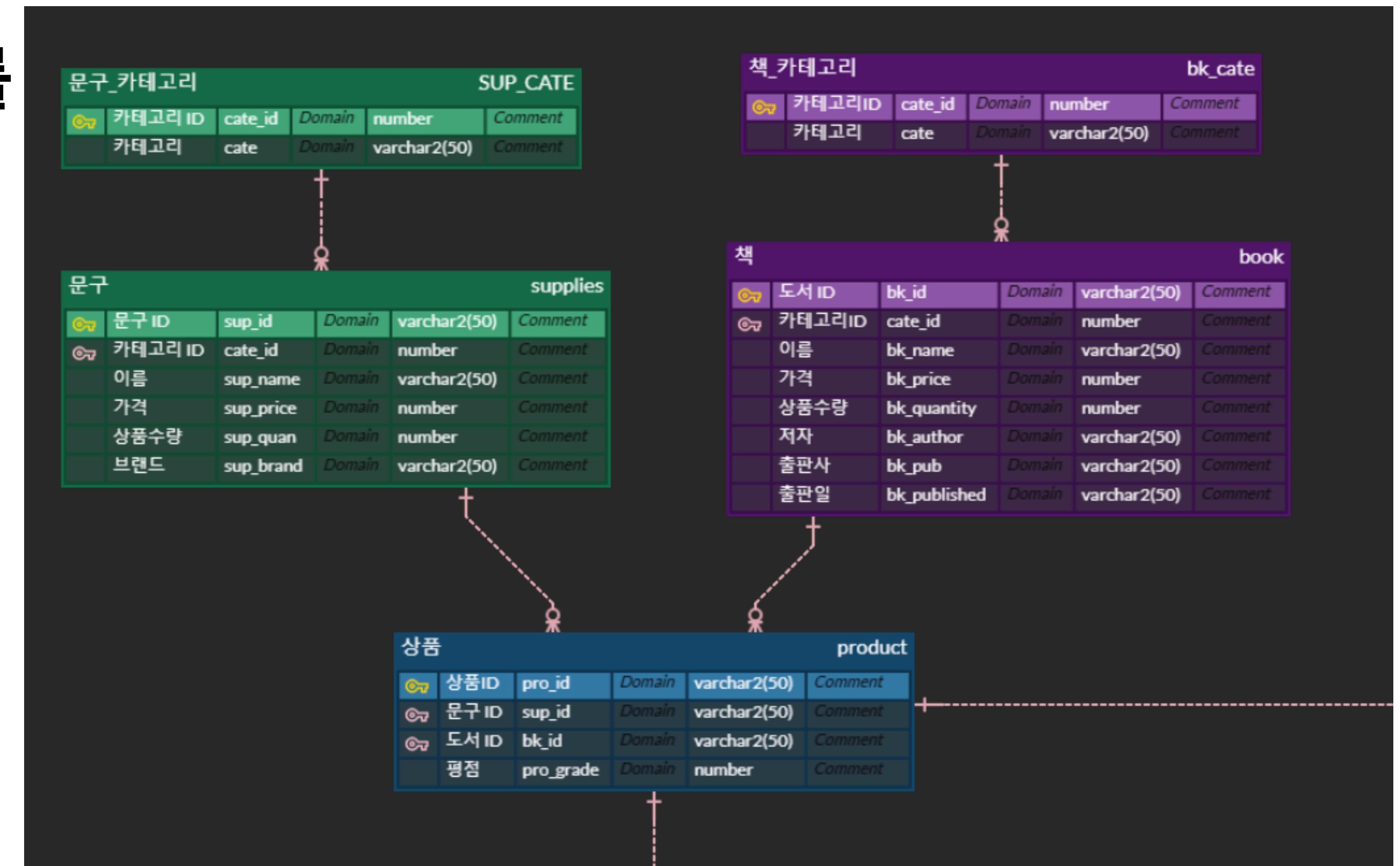
문구/GIFT
+ 예스굿즈
- 디자인문구
+ <a href="#">다이어리/스케줄러</a>
+ <a href="#">고급비즈니스 문구</a>
+ <a href="#">스터디플래너/컨셉북</a>
+ 캘린더
+ 노트
+ 파일/바inder
+ 메모용품
+ 사무/학습용품
+ 스티커/테잎
+ 카드/편지/엽서
+ 선물포장용품
+ 데스크용품/지구본
문구세트
기타문구
+ 필기구
+ 고급필기구/만년필
+ 미술용품
필통/파우치



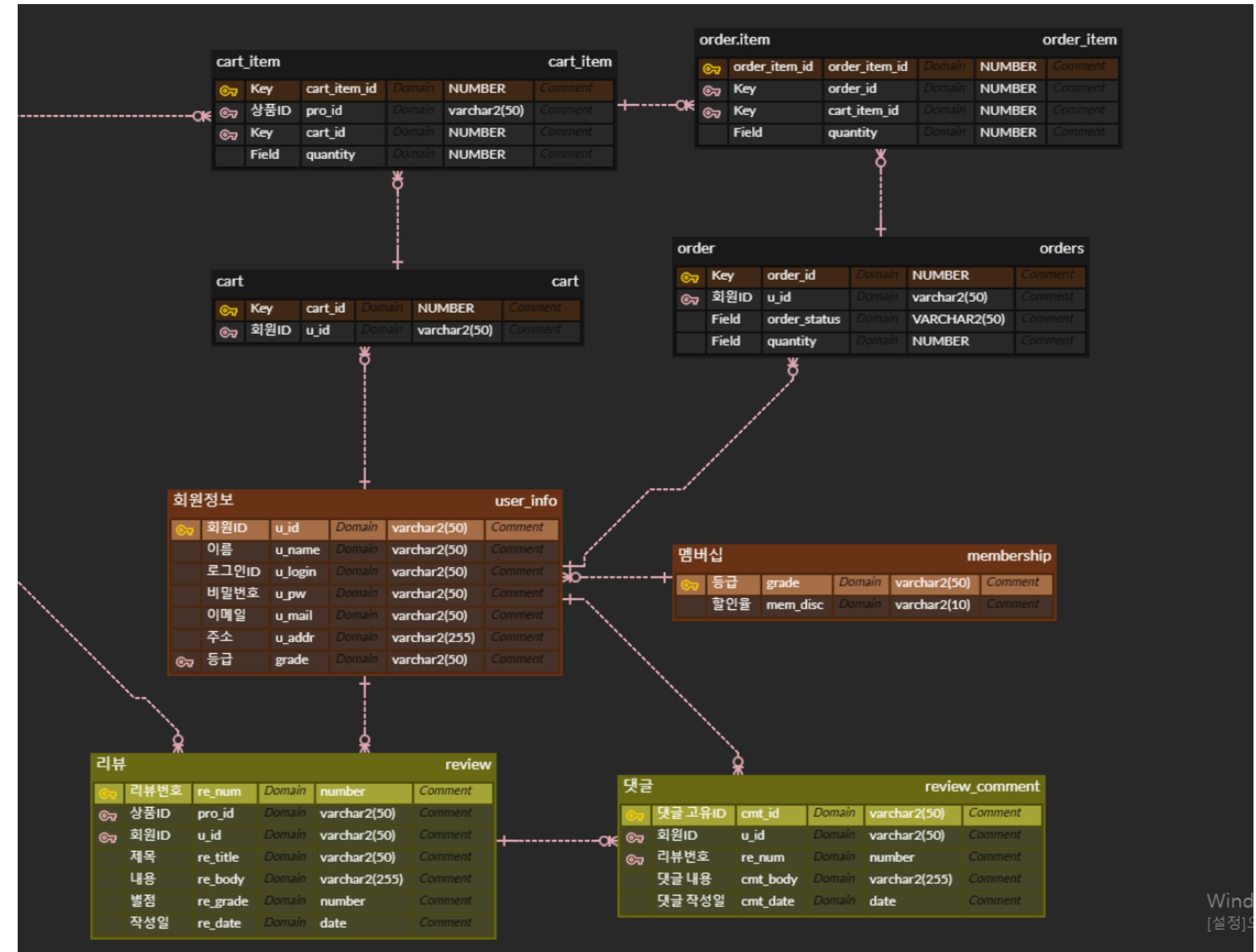
# ERD 클라우드를 이용한 테이블 관계도



# ERD 클라우드를 이용한 테이블 관계도



# ERD 클라우드를 이용한 테이블 관계도



# 시스템 흐름도

## [회원]

1. 로그인 → 2. 장바구니 담기 → 3. 주문 생성 → 4. 결제

## [시스템]

1. 재고 차감 → 2. 할인 적용 → 3. 결제 기록 → 4. 매출 반영

## [관리자]

1. 매출 확인 → 2. 재고 보충 → 3. 리뷰 모니터링

# 테이블 명세서

SUP_CATE							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	카테고리 ID	cate_id		number	N		
	카테고리	cate		varchar2(50)	Y		
review_comment							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	댓글 고유ID	cmt_id		varchar2(50)	N		
FK	회원ID	u_id		varchar2(50)	N		
FK	리뷰번호	re_num		number	N		
	댓글 내용	cmt_body		varchar2(255)	Y		
	댓글 작성일	cmt_date		date	Y		
book							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	도서 ID	bk_id		varchar2(50)	N		
FK	카테고리ID	cate_id		number	N		
	이름	bk_name		varchar2(50)	N		
	가격	bk_price		number	N		
	상품수량	bk_quantity		number	N		
	저자	bk_author		varchar2(50)	N		
	출판사	bk_pub		varchar2(50)	N		
	출판일	bk_published		varchar2(50)	N		

order_item							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	order_item_id	order_item_id		NUMBER	N		
FK	Key	order_id		NUMBER	N		
FK	Key	cart_item_id		NUMBER	N		
	Field	quantity		NUMBER	Y		
cart							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	Key	cart_id		NUMBER	N		
FK	회원ID	u_id		varchar2(50)	N		
user_info							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	회원ID	u_id		varchar2(50)	N		
	이름	u_name		varchar2(50)	Y		
	로그인ID	u_login		varchar2(50)	Y		
	비밀번호	u_pw		varchar2(50)	Y		
	이메일	u_mail		varchar2(50)	Y		
	주소	u_addr		varchar2(255)	Y		
FK	등급	grade		varchar2(50)	N		

# 테이블 명세서

product							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	상품ID	pro_id		varchar2(50)	N		
FK	문구 ID	sup_id		varchar2(50)	Y		
FK	도서 ID	bk_id		varchar2(50)	Y		
	평점	pro_grade		number	Y		
orders							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	Key	order_id		NUMBER	N		
FK	회원ID	u_id		varchar2(50)	N		
	Field	order_status		VARCHAR2(50)	Y		
	Field	quantity		NUMBER	Y		
review							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	리뷰번호	re_num		number	N		
FK	상품ID	pro_id		varchar2(50)	N		
FK	회원ID	u_id		varchar2(50)	N		
	제목	re_title		varchar2(50)	Y		
	내용	re_body		varchar2(255)	Y		
	별점	re_grade		number	Y		
	작성일	re_date		date	Y		

cart_item							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	Key	cart_item_id		NUMBER	N		
FK	상품ID	pro_id		varchar2(50)	N		
FK	Key	cart_id		NUMBER	N		
	Field	quantity		NUMBER	Y		
supplies							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	문구 ID	sup_id		varchar2(50)	N		
FK	카테고리 ID	cate_id		number	N		
	이름	sup_name		varchar2(50)	Y		
	가격	sup_price		number	Y		
	상품수량	sup_quan		number	Y		
	브랜드	sup_brand		varchar2(50)	Y		
membership							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	등급	grade		varchar2(50)	N		
	할인율	mem_disc		varchar2(10)	Y		
bk_cate							
Key	Logical	Physical	Domain	Type	Allow Null	Default Value	Comment
PK	카테고리ID	cate_id		number	N		
	카테고리	cate		varchar2(50)	Y		

# 테이블 생성

## PK/FK 관계 정의

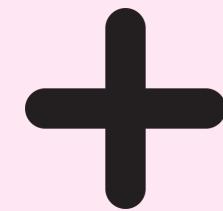
```
DROP TABLE user_info;

CREATE TABLE user_info (
    u_id      varchar2(50)          NOT NULL,
    u_name    varchar2(50)          NULL,
    u_login   varchar2(50)          NULL,
    u_pw      varchar2(50)          NULL,
    u_mail    varchar2(50)          NULL,
    u_addr    varchar2(255)         NULL,
    grade     varchar2(50)          NOT NULL
);

ALTER TABLE user_info ADD CONSTRAINT PK_USER_INFO PRIMARY KEY (
    u_id
);

ALTER TABLE review_comment ADD CONSTRAINT FK_user_info_TO_review_comment_1 FOREIGN KEY (
    u_id
)
REFERENCES user_info (
    u_id
);
```

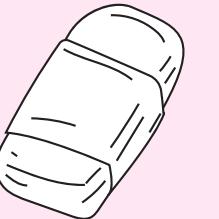
# 데이터 추가, 검색, 삭제



```
insert all  
--bk_cate테이블에 데이터 입력--  
  into bk_cate    values(1, 'IT')  
  into bk_cate    values(2, '소설')  
  into bk_cate    values(3, '자기계발')
```



```
select cate_id "카테고리",  
       bk_id "책번호",  
       sum(bk_quantity) "재고량",  
       sum(bk_price) "총합계금액" from book  
group by cate_id, bk_id order by cate_id, bk_id;
```



```
drop table bk_cate CASCADE CONSTRAINTS purge;  
drop table book CASCADE CONSTRAINTS purge;  
drop table cart CASCADE CONSTRAINTS purge;  
drop table cart_item CASCADE CONSTRAINTS purge;
```

# 데이터 추가, 검색, 삭제

```
-- 할인율이 10% 이하인 고객의 이름, 이메일, 주소, 등급, 할인율 출력 -> 등급으로 내림차순 정렬  
create or replace view cust_grade as  
select u.u_name "고객명", u.u_mail "이메일", u.u_addr "고객주소", m.grade "등급",  
       to_number(replace(mem_disc, '%', '')) "할인율"  
  from user_info u, membership m  
 where u.grade=m.grade and  
       to_number(replace(mem_disc, '%', '')) <= 10 order by u.grade desc;  
  
select * from cust_grade;
```

SQL | 인출된 모든 행: 34(0,008초)

#	고객명	이메일	고객주소	등급	할인율
1	송지수	song321@example.com	경기도 안양시	Silver	10
2	송윤희	song3217@example.com	서울특별시 성북구	Silver	10
3	김민주	kim910@example.com	경기도 성남시	Silver	10
4	정윤미	jeong987@example.com	서울특별시 서초구	Silver	10
5	이시은	lee159@example.com	대전광역시 서구	Silver	10
6	박민석	park987@example.com	대구광역시 중구	Silver	10
7	정수빈	jeong456@example.com	광주광역시 북구	Silver	10
8	박수진	park543@example.com	경상북도 포항시	Silver	10
9	이호준	lee987@example.com	충청북도 청주시	Silver	10
10	정다영	jeong432@example.com	광주광역시 서구	Silver	10
11	오세훈	ohsehun123@example.com	경기도 수원시	Silver	10
12	김도현	kim321@example.com	부산광역시 동래구	Silver	10
13	유진영	yujin567@example.com	경기도 화성시	Silver	10
14	박준형	park567@example.com	전라남도 목포시	Silver	10
15	이하늘	sky456@example.com	부산광역시 해운대구	Silver	10

# 데이터 추가, 검색, 삭제

--2. 주소가 서울인 고객의 이름, 주소, 등급, 할인율 출력 ->이름으로 오름차순 정렬

```
SELECT u.u_name, u.u_addr, m.grade, m.mem_disc
FROM user_info u, membership m
WHERE u.grade = m.grade
AND u.u_addr LIKE '서울%'
ORDER BY u.u_name;
```

질의 결과 X

SQL | 인출된 모든 행: 14(0.002초)

	U_NAME	U_ADDR	GRADE	MEM_DISC
1	김보영	서울특별시 강북구	Gold	15
2	김영민	서울특별시 강남구 역삼동	Bronze	5
3	박진영	서울특별시 송파구	Bronze	5
4	송윤희	서울특별시 성북구	Silver	10
5	유다혜	서울특별시 구로구	Gold	15
6	이민지	서울특별시 성동구	Gold	15
7	이소희	서울특별시 종로구	Gold	15
8	이정은	서울특별시 용산구	Gold	15
9	정다영	서울특별시 강동구	Silver	10
10	정유진	서울특별시 강남구	Bronze	5
11	정윤미	서울특별시 서초구	Silver	10

# 데이터 검색 - 총 매출 조회, 월별 매출 분석

```
-- 1. 전체 총 매출 조회
```

```
SELECT SUM(payment_amount) AS "총 매출"  
FROM payment;
```

	□ "총 매출" ▾	+
1	49590	

```
-- 2. 월별 매출 분석
```

```
SELECT TO_CHAR(payment_date, 'YYYY-MM') AS "년-월",  
       SUM(payment_amount)                   AS "월별 매출"  
FROM payment  
GROUP BY TO_CHAR(payment_date, 'YYYY-MM')  
ORDER BY "년-월";
```

	□ "년-월" ▾	+	□ "월별 매출" ▾	+
1	2025-04			49590

# 데이터 검색 - 상품 카테고리별 매출, 회원 등급별 매출

-- 3. 상품 카테고리별 매출

```
SELECT CASE
    WHEN p.pro_id LIKE 'BK_%' THEN '도서'
    WHEN p.pro_id LIKE 'SUP_%' THEN '문구'
    END AS "카테고리",
    SUM(oi.unit_price * oi.quantity) AS "매출"
FROM order_item oi
JOIN product p [1..n<->1] ON oi.pro_id = p.pro_id
GROUP BY CASE
    WHEN p.pro_id LIKE 'BK_%' THEN '도서'
    WHEN p.pro_id LIKE 'SUP_%' THEN '문구'
    END;
```



	"카테고리"	"매출"
1	도서	46000
2	문구	6200

-- 4. 회원 등급별 매출

```
SELECT u.grade AS "회원 등급",
       SUM(p.payment_amount) AS "매출"
FROM payment p
JOIN orders o [1..n<->1] ON p.order_id = o.order_id
JOIN user_info u [1..n<->1] ON o.u_id = u.u_id
GROUP BY u.grade;
```

	"회원 등급"	"매출"
1	Bronze	49590

# **데이터 활용**

# **뷰/함수/프로시저/트리거**

# 뷰 1 - 문구관리

-- 각 브랜드별 등록된 문구의 개수

```
create or replace view cnt_sup_view as
  select (count(s.sup_brand)||'개') "등록된 브랜드 수", s.sup_brand "브랜드명"
    from supplies s, sup_cate sc
   where sc.cate_id = s.cate_id
  group by s.sup_brand
 order by count(s.sup_brand);
```

-- 카테고리가 메모용품인 문구의 재고합과 필기구인 문구의 재고합의 총 합계

```
create or replace view sum_sup_view as
  select
    (select sum(sup_quan)
      from supplies s
      join sup_cate sc on s.cate_id=sc.cate_id
     where sc.cate='메모용품') as "메모용품 재고",
    (select sum(sup_quan)
      from supplies s
      join sup_cate sc on s.cate_id=sc.cate_id
     where sc.cate='필기구') as "필기구 재고",
    (select sum(s.sup_quan)
      from supplies s
      join sup_cate sc on s.cate_id = sc.cate_id
     where sc.cate in ('메모용품', '필기구')) as "합계 재고"
  from dual;
```

## 브랜드별 등록된 문구의 개수 카테고리별 각 문구의 재고합, 총 재고합

#	등록된 브랜드 수	브랜드명
1	1개	칠삼이일
2	1개	스터디메이트
3	1개	신한화구
4	1개	파카
5	1개	3M

```
select * from cnt_sup_view;
select * from sum_sup_view;
```

#	메모용품 재고	필기구 재고	합계 재고
1	870	410	1280

## 뷰 2 - 도서관리

```
-- 뷰 연습 --
GRANT create view to bookstore;

create or replace view myView as
select bk_name "책이름", bk_id "도서번호",
       (select cate_id from book where b.bk_id=bk_id) "카테고리번호",
       (select bk_price from book where b.bk_id=bk_id) "책가격"
  from book b
 where
cate_id=(select cate_id from book where bk_name='파이썬');

SELECT 책이름, 도서번호 FROM myView WHERE 책이름 = '리액트 입문';
SELECT 책이름, 도서번호, 카테고리번호, 책가격 FROM myView WHERE 책이름 = '파이썬';
```

	◆ 책이름	◆ 도서번호	◆ 카테고리번호	◆ 책가격
1	파이썬	b2		1 25000

## 부 3-1 일별 매출 합계

-- 1. 일별 매출 뷰

```
CREATE VIEW daily_sales AS
SELECT TRUNC(payment_date) AS "판매일",
       SUM(payment_amount) AS "일매출"
FROM payment
GROUP BY TRUNC(payment_date);

select * from daily_sales;
```

	□ 판매일 ↴	▪	□ 일매출 ↴	▪
1	2025-04-01		49590	

## 뷰 3-2 베스트셀러

```
-- 2. 베스트셀러 상품 뷰  
CREATE VIEW bestseller AS  
SELECT p.pro_id,  
       SUM(oi.quantity)           AS "총 판매량",  
       SUM(oi.unit_price * oi.quantity) AS "매출"  
FROM order_item oi  
      JOIN product p ON oi.pro_id = p.pro_id  
GROUP BY p.pro_id  
ORDER BY "총 판매량" DESC;
```

	□ PRO_ID ▼	□ 총 판매량 ▼	□ 매출 ▼
1	BK_B1	2	46000
2	SUP_S1	1	6200

# 뷰 3-3 회원 등급별 구매 집계

```
-- 3. 회원 구매 분석 뷰
CREATE OR REPLACE VIEW member_purchase_analysis AS
SELECT
    u.u_id,
    u.u_name,
    m.grade,
    m.mem_disc,
    COUNT(DISTINCT o.order_id) AS total_orders,
    SUM(p.payment_amount) AS total_spent,
    ROUND(SUM(p.payment_amount) / COUNT(DISTINCT o.order_id), 2) AS avg_order_value
FROM user_info u
    JOIN membership m [1..n<->1..1] ON u.grade = m.grade
    JOIN orders o [1<->1..n] ON u.u_id = o.u_id
    JOIN payment p [1<->1..n] ON o.order_id = p.order_id
GROUP BY u.u_id, u.u_name, m.grade, m.mem_disc
ORDER BY total_spent DESC;
```

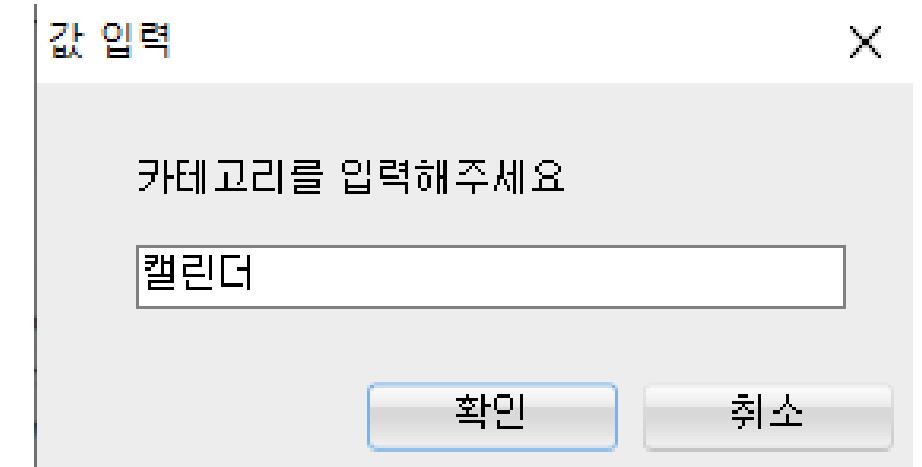
	□ U_ID □	□ U_NAME □	□ GRADE □	□ MEM_DISC □	□ TOTAL_ORDERS □	□ TOTAL_SPENT □	□ AVG_ORDER_VALUE □
1	A0001	김영민	Bronze	5%	1	49590	49590

# 프로시저 1 - 카테고리 입력 시 가장 비싼 상품 출력

```
-- 1. 카테고리를 입력하면 해당 브랜드의 가장 값이 비싼 제품을 출력하는 프로시저
CREATE OR REPLACE PROCEDURE get_most_exp_pro (
    p_cate_name IN sup_cate.cate%TYPE      -- 입력: 카테고리명
) IS
    v_sup_n supplies.sup_name%TYPE;        -- 출력: 제품명
    v_sup_pri supplies.sup_price%TYPE;      -- 출력: 가격
    v_sup_b supplies.sup_brand%TYPE;        -- 출력: 브랜드명
-- 일반 변수 커서 설정
    cursor most_exp IS
        select sup_name, sup_price, sup_brand
        from supplies s
        join sup_cate sc on sc.cate_id = s.cate_id
        where sc.cate = p_cate_name
        order by s.sup_price desc
        fetch first 1 row only; -- 내림차열 첫번째 행 값 select
begin
-- 커서 열기
    open most_exp;
    fetch most_exp into v_sup_n, v_sup_pri, v_sup_b;
    IF most_exp%NOTFOUND THEN
        -- 데이터가 없을 경우
        DBMS_OUTPUT.PUT_LINE('입력한 카테고리에 해당하는 제품이 없습니다.');
    ELSE
        -- 데이터가 있을 경우
        DBMS_OUTPUT.PUT_LINE('가장 비싼 제품: ' || v_sup_n || '(' || v_sup_b || ')');
        DBMS_OUTPUT.PUT_LINE('가격: ' || TO_CHAR(v_sup_pri, 'L00,000'));
    END IF;
    close most_exp;
end;
/
```

--프로시저

```
accept sup_cn prompt '카테고리를 입력해주세요'
exec get_most_exp_pro('&sup_cn'); -- 캘린더, 노트 등 입력
```

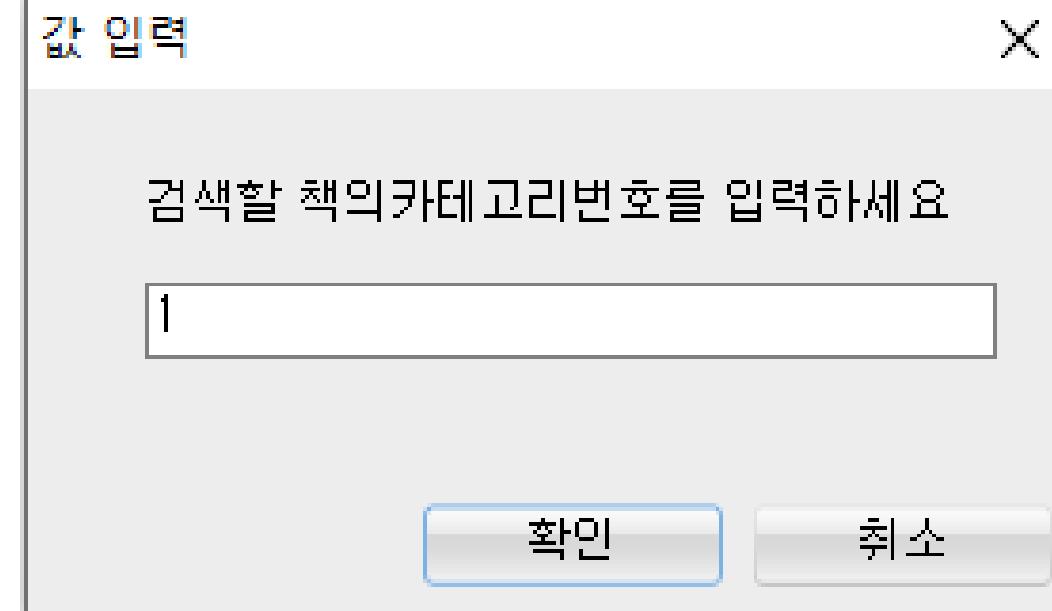


가장 비싼 제품: 2025 벽걸이 캘린더 (디자인공간)  
가격: ₩12,000

PL/SQL 프로시저가 성공적으로 완료되었습니다.

# 프로시저 2 - 카테고리 번호를 통해 도서 정보를 출력

```
128 --3. 책의 카테고리 번호를 매개변수로 입력받아 도서번호, 책이름, 저자, 출판일 가격을 출력하는 프로시저
129 select bk_id, bk_name, book.bk_author, bk_published, bk_price from book
130 where cate_id=4;
131
132 create or replace procedure book_samecate_id_p (
133     p_bkpcate_id in book.cate_id%type
134 ) is
135     -- 일반 변수 선언 영역
136 begin
137     for book_buf in (select bk_id, bk_name, book.bk_author, bk_published, bk_price
138         from book where cate_id=p_bkpcate_id)
139     loop
140         dbms_output.put_line('도서번호: ' || book_buf.bk_id);
141         dbms_output.put_line('책이름: ' || book_buf.bk_name);
142         dbms_output.put_line('저자: ' || book_buf.bk_author);
143         dbms_output.put_line('출판일: ' || book_buf.bk_published);
144         dbms_output.put_line('책가격: ' || book_buf.bk_price);
145         dbms_output.put_line('-----');
146     end loop;
147 end;
148 /
149 accept p_cate_id prompt '검색할 책의 카테고리번호를 입력하세요'
150 exec book_samecate_id_p(&p_cate_id);
```



도서번호: b1  
책이름: 오라클SQL  
저자: 이지훈  
출판일: 2018/10/30  
책가격: 23000  
-----  
도서번호: b2  
책이름: 파이썬  
저자: 김영희  
출판일: 2018/10/31  
책가격: 25000

# 프로시저 3 - 회원 등급에 따라 주문 총액 할인 적용

```
CREATE OR REPLACE PROCEDURE apply_discount (
    p_order_id IN NUMBER,
    p_user_id IN VARCHAR2
) AS
    v_discount_rate NUMBER := 0;
    v_total_amount NUMBER := 0;
    v_mem_disc VARCHAR2(10);
BEGIN
    BEGIN
        SELECT mem_disc INTO v_mem_disc
        FROM membership m
        JOIN user_info u ON m.grade = u.grade
        WHERE u.u_id = p_user_id;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            v_mem_disc := '0%';
    END;

    BEGIN
        v_discount_rate := TO_NUMBER(REPLACE(v_mem_disc, '%', '')) / 100;
    EXCEPTION
        WHEN VALUE_ERROR THEN
            v_discount_rate := 0;
    END;

    SELECT SUM(unit_price * quantity) INTO v_total_amount
    FROM order_item
    WHERE order_id = p_order_id;

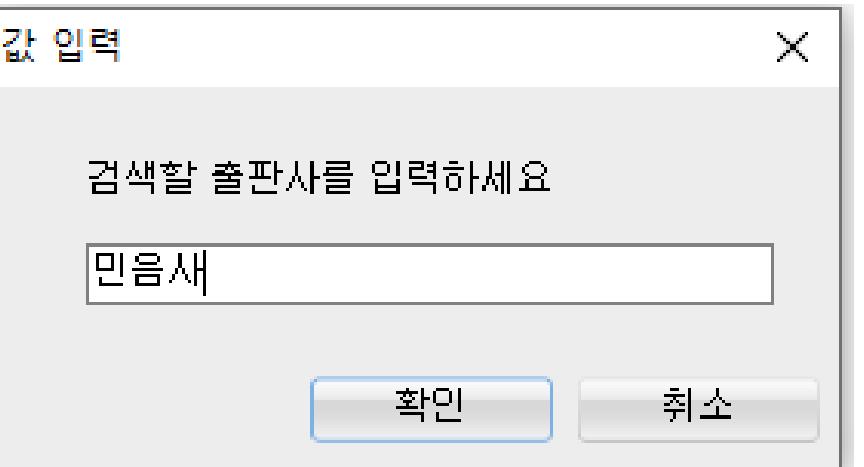
    UPDATE orders
    SET total_amount = NVL(v_total_amount, 0) * (1 - NVL(v_discount_rate, 0))
    WHERE order_id = p_order_id;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

user\_info → membership →  
할인율 계산 → orders.total\_amount 업데이트

# 프로시저 4 - 출판사를 입력하여 가장 비싼 책과 가격 을 검색

```
create or replace procedure bk_maxprice_p (
    p_bkpub in book.bk_pub%type
) is
    v_bk_price book.bk_price%type;
    v_bk_name book.bk_name%type;
begin
    select bk_price, bk_name into v_bk_price, v_bk_name
        from book where bk_pub=p_bkpub
        and bk_price =(select max(bk_price) from book where bk_pub=p_bkpub);
    dbms_output.put_line
        (p_bkpub || '출판사의 도서 ' || v_bk_name || '의 최대가격:' || v_bk_price);
END;
/
accept p_bkpublic prompt '검색할 출판사를 입력하세요'
exec bk_maxprice_p('&p_bkpublic');
```



PL/SQL 프로시저가 성공적으로 완료되었습니다.

민음사출판사의 도서 UI/UX 디자인의 최대가격: 44000

PL/SQL 프로시저가 성공적으로 완료되었습니다.

# 트리거 1 - 평점관리

- 2. pro\_grade의 값이 review 테이블의
- pro\_id의 데이터에 연결된 각 re\_grade의 평균이 되도록 자동으로 반영

```
CREATE OR REPLACE TRIGGER trg_after_update_grade
    AFTER INSERT OR UPDATE OR DELETE ON review
DECLARE
    -- 변경된 pro_id 목록 저장
    CURSOR cur_pro_id IS
        SELECT DISTINCT pro_id FROM review;

BEGIN
    -- product 테이블 업데이트
    FOR rec IN cur_pro_id LOOP
        UPDATE product p
        SET p.pro_grade =
            (SELECT NVL(AVG(re_grade), 0)
             FROM review
             WHERE pro_id = rec.pro_id)
        WHERE p.pro_id = rec.pro_id;
    END LOOP;
END;
```

/

```
insert into review VALUES (13,'BK_b1','A0020','데이터 시각화','재밌습니다.',8,
insert into review VALUES (14,'BK_b1','A0020','데이터 시각화','재밌습니다.',7,
insert into review VALUES (15,'BK_b1','A0020','데이터 시각화','재밌습니다.',9,
insert into review VALUES (16,'BK_b1','A0020','데이터 시각화','재밌습니다.',10,
insert into review VALUES (17,'BK_b1','A0020','데이터 시각화','재밌습니다.',3,
```

→ Review 테이블에 데이터를 추가

RE_NUM	PRO_ID	U_ID	RE_TITLE	RE_BODY	RE_GRADE
13	BK_b1	A0020	데이...	재밌...	9
14	BK_b1	A0020	데이...	재밌...	7
15	BK_b1	A0020	데이...	재밌...	9
16	BK_b1	A0020	데이...	재밌...	10
17	BK_b1	A0020	데이...	재밌...	3

PRO\_ID가 같은 상품의 각 평점 평균을 →

PRO_ID	SUP_ID	BK_ID	PRO_GRADE
1 BK_b1	(null)	b1	7.6

→ Product 테이블의 평점에 반영

# 트리거 2 - Product 테이블 관리, 재고 관리

```
-- 카운트 트리거
CREATE OR REPLACE TRIGGER trg_after_insert_book
    AFTER INSERT ON book
    FOR EACH ROW
BEGIN
    INSERT INTO product (pro_id, bk_id, sup_id, pro_grade)
        VALUES ('BK_' || :NEW.bk_id, :NEW.bk_id, NULL, NULL);
END;
/
CREATE OR REPLACE TRIGGER trg_after_insert_supplies
    AFTER INSERT ON supplies
    FOR EACH ROW
BEGIN
    INSERT INTO product (pro_id, sup_id, bk_id, pro_grade)
        VALUES ('SUP_' || :NEW.sup_id, :NEW.sup_id, NULL, NULL);
END;
/
commit;
```

BOOK, SUPPLIES 테이블에 INSERT 이벤트 발생 시,  
PRODUCT 테이블에 항목 추가

```
CREATE OR REPLACE TRIGGER trg_after_insert_order_item
AFTER INSERT ON order_item
FOR EACH ROW
DECLARE
    v_sup_id VARCHAR2(50);
    v_bk_id VARCHAR2(50);
BEGIN
    SELECT sup_id, bk_id INTO v_sup_id, v_bk_id
    FROM product
    WHERE pro_id = :NEW.pro_id;

    IF v_sup_id IS NOT NULL THEN
        UPDATE supplies
        SET sup_quan = sup_quan - :NEW.quantity
        WHERE sup_id = v_sup_id;
    ELSIF v_bk_id IS NOT NULL THEN
        UPDATE book
        SET bk_quantity = bk_quantity - :NEW.quantity
        WHERE bk_id = v_bk_id;
    END IF;
END;
/
```

주문 완료 시 재고(수량) 차감  
(book 또는 supplies 테이블 업데이트)

Thank you

Q&A