250715 SQL로 추천 시스템 설계하 기(프로젝트)

데이터셋 링크: Amazon Sales Dataset

필드명	설명						
product_id	제품의 고유 식별자						
product_name	제품의 이름						
category	제품이 속한 카테고리 정보						
discounted_price	할인된 가격						
actual_price	정가						
discount_percentage	할인율						
rating	제품의 평균 평점						
rating_count	제품에 대한 총 평점 수						
about_product	제품에 대한 간단한 설명						
user_id	리뷰를 작성한 사용자의 고유 식별자 (쉼표로 구분된 여러 사용자)						
user_name	리뷰를 작성한 사용자 이름 (쉼표로 구분된 여러 사용자)						
review_id	리뷰의 고유 식별자 (쉼표로 구분된 여러 리뷰)						
review_title	리뷰 제목 (쉼표로 구분된 여러 리뷰 제목)						
review_content	리뷰 내용 (쉼표로 구분된 여러 리뷰 내용)						
img_link	제품 이미지 URL						
product_link	제품 페이지 URL						

데이터 정제

-- 결측치 확인

SELECT COUNT(*) cnt_all, COUNT(product_id) cnt_product_id, COUNT(product_name) cnt_product_name, COUNT(category) cnt_category,

COUNT(discounted_price) cnt_discounted_price,

COUNT(actual_price) cnt_actual_price,

COUNT(discount_percentage) cnt_discount_percentage,

COUNT(rating) cnt_rating, COUNT(rating_count) cnt_rating_count,

COUNT(about_product) cnt_about_product, COUNT(user_id) cnt_user_i d,

COUNT(user_name) cnt_user_name, COUNT(review_id) cnt_review_id, COUNT(review_title) cnt_review_title,

COUNT(review_content) cnt_review_content

FROM 'our-highway-465502-r0.project.amazon';



- -- rating_count 컬럼에만 결측치가 2건 있음
- -- 결측행, 사용하지 않을 컬럼(img_link, product_link) 제거 CREATE OR REPLACE TABLE `our-highway-465502-r0.project.amazon` AS

SELECT

* EXCEPT (img_link, product_link)

FROM 'our-highway-465502-r0.project.amazon'

WHERE rating_count IS NOT NULL;

● 이 문으로 이름이 amazon인 테이블이 교체되었습니다.

-- 중복값 확인하기

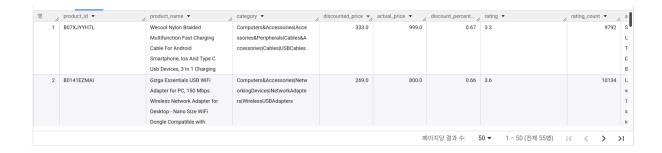
SELECT *

FROM 'our-highway-465502-r0.project.amazon'

GROUP BY product_id, product_name, category, discounted_price, actual_price, discount_percentage, rating, rating_count, about_product, user_id, user_name,

review_id, review_title, review_content HAVING COUNT(*) > 1;

-- 중복값이 있음을 확인할 수 있음



- -- 중복값이 있다고 나온 'B07XJYYH7L' 상품을 조회해보면 상품 설명, 카테고리 부터 평점까지 모든 값이 같게 나왔음
- --> 같은 데이터가 중복으로 잘못 들어간 것이라고 생각해 제거하기로 결정함.

SELECT*

FROM 'our-highway-465502-r0.project.amazon'

WHERE product_id = 'B07XJYYH7L';



-- 중복값 처리하기

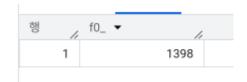
CREATE OR REPLACE TABLE `our-highway-465502-r0.project.amazon` AS SELECT DISTINCT *

FROM 'our-highway-465502-r0.project.amazon';

-- 중복 제거 후 데이터 개수

SELECT COUNT(*)

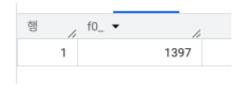
FROM 'our-highway-465502-r0.project.amazon';



오류값 처리

```
-- rating 컬럼의 오류값 처리
-- rating의 max 값을 찍어보면 '|' 값이 나옴
SELECT max(rating)
FROM `our-highway-465502-r0.project.amazon`;

-- '|' 값이 들어있는 행 삭제하기
DELETE FROM `our-highway-465502-r0.project.amazon`
WHERE rating = '|';
SELECT COUNT(*)
FROM `our-highway-465502-r0.project.amazon`;
```



추천 시스템 1: 후기가 넘치는 인기 제품 추천

컨셉: 리뷰 수가 많다는 건 많은 소비자가 직접 사용해봤고, 피드백을 남길 만큼 만족하거나 불만이 있었다는 뜻이므로, 리뷰 수와 평점을 함께 고려해, **다수 사용자에게 인정받은 인기** 제품을 추천

평점이 4.5 이상이면서 rating_count, review_count 가 높은 제품 추천

핵심 질문: "많은 소비자들이 실제로 사용하고 평가한 인기 제품은 무엇일까?"

주요 컬럼: rating_count , rating , review_count

SQL 로직:

-- 2. 후기가 넘치는 인기 제품 추천 : rating이 4.5 이상이고, rating_count, review_count 둘다 높은 상품 SELECT *,

ARRAY_LENGTH(SPLIT(user_id, ',')) AS review_count FROM `our-highway-465502-r0.project.amazon`

WHERE CAST(rating AS FLOAT64) >= 4.5 ORDER BY rating_count DESC, review_count DESC;



추천 시스템 2 : 가격 대비 만족도 추천 시스템 (가성비 우수 상 품)

컨셉: 정가와 할인 가격의 차이만 보는 것이 아니라, **최종 구매가 대비 얼마나 만족했는지를** 평가

예를 들어, **비싼 제품이 평점 4.2를 받는 것보다, 저렴한 가격에 4.5를 받은 제품**이 더 만족도 높은 가성비 제품일 수 있으므로, "이 정도 가격이면 이 정도 품질이면 만족이다!"라는 관점 의 추천

핵심 질문: "실제 구매 가격을 고려했을 때, 고객 만족도가 가장 높은 제품은 무엇일까?"

주요 컬럼: discounted_price , rating , rating_count

SQL 로직:

```
-- 1. 가격 대비 만족도 추천 시스템 (가성비 우수 상품) : 최종 구매가 대비 평점 SELECT product_id, product_name, rating, discounted_price, SAFE_DIVIDE(CAST(rating AS FLOAT64), discounted_price) AS value_scor e FROM `our-highway-465502-r0.project.amazon` WHERE rating_count >= 10 -- 신뢰도 보완용
```

ORDER BY value_score DESC LIMIT 20;



추천 시스템 3: 카테고리별 인기 제품 추천

컨셉: 고객의 관심사(카테고리)에 따라 맞춤형으로 평점이 높은 인기 제품을 추천해주는 시스템.

같은 제품이라도 관심 분야가 다르면 추천 우선순위도 달라지므로, **카테고리별로 평점과 평점 수가 높은 제품**을 추려 보여줌

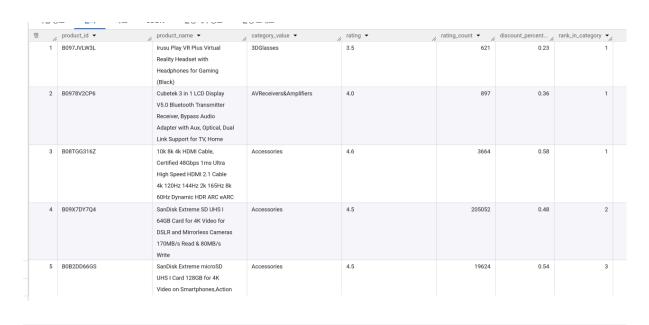
핵심 질문: "이 카테고리에서 지금 가장 인기 있는 제품은 무엇일까?"

주요 컬럼: category , rating , rating_count

SQL 로직:

-- 3. 카테고리별 베스트셀러 추천 : 카테고리별로 평점과 리뷰 수가 높은 제품 WITH ranked_products AS (
SELECT product_id, product_name, category_value,

```
rating,
  rating_count,
  discount_percentage,
  -- category_value별로 rating, rating_count가 높은 순으로 순위 매기기
  RANK() OVER (
   PARTITION BY category_value
   ORDER BY rating DESC, rating_count DESC
  ) AS rank_in_category
 FROM (SELECT *
    FROM 'our-highway-465502-r0.project.amazon',
        -- category 필드를 버티컬 라인())을 기준으로 나누기
      UNNEST(SPLIT(category, '|')) AS category_value)
)
SELECT*
FROM ranked_products
-- 각 카테고리별로 순위가 3위인 것 까지만 필터링
WHERE rank_in_category <= 3
ORDER BY category_value, rank_in_category;
```



추천 시스템 4 : 신뢰할 수 있는 유저들이 고른 제품

컨셉: 리뷰를 많이 남긴 **활동적인 유저들**은 경험이 많고 신뢰도가 높은 편이므로, 이들이 여러 번 추천한 제품은 자연스럽게 신뢰할 수 있는 제품이 될 수 있음.

이러한 **고빈도 리뷰어들의 교집합**을 찾아 추천

핵심 질문: "리뷰를 자주 남기는 유저들이 공통적으로 추천한 제품은 무엇일까?"

주요 컬럼: user_id , review_id , product_id

SQL 로직:

```
-- 4. 신뢰할 수 있는 유저들이 고른 제품 : user_id 별 리뷰 수를 집계한 뒤, 많이 작성
한 유저를 기준으로 필터링하여 해당 유저들이 공통적으로 리뷰한 제품의 출현 빈도를
계산해 상위 제품을 추려내기
WITH temp_table AS (
 SELECT *
 FROM 'our-highway-465502-r0.project.amazon',
   UNNEST(SPLIT(user_id, ',')) AS user_id_split
)
SELECT *
FROM 'our-highway-465502-r0.project.amazon'
WHERE product_id IN (SELECT product_id
          FROM temp_table
          WHERE user_id_split IN (SELECT user_id_split
                      FROM temp_table
                      GROUP BY user_id_split
                      HAVING COUNT(*) >= 5)
          GROUP BY product_id
          HAVING COUNT(*) >= 10)
```

3 /	product_id ▼	product_name •	category •	discounted_price */	actual_price ▼ //	discount_percent	rating ▼	rating_count -	about_product ▼	user_id ▼
1	B0B5B6PQCT	boAt Wave Call Smart Watch,	Electronics WearableTechnolog	1799.0	7990.0	0.77	3.8	17833	Bluetooth Calling- Wave Call	AHPYDFW6Y3FIQ
		Smart Talk with Advanced							comes with a premium built-in	5QNVRQ,AG7DTV
		Dedicated Bluetooth Calling							speaker and bluetooth calling	6V2G4KSIB67TDA
		Chip, 1.69" HD Display with							via which you can stay	4I23HHMRHCCCI7
		550 NITS & 70% Color Gamut,							connected to your friends,	HPOQQONRLZMH
2	B0B5B6PQCT	boAt Wave Call Smart Watch,	Electronics WearableTechnolog	1999.0	7990.0	0.75	3.8	17831	Bluetooth Calling- Wave Call	AHPYDFW6Y3FIQ
		Smart Talk with Advanced							comes with a premium built-in	5QNVRQ,AG7DTV
		Dedicated Bluetooth Calling							speaker and bluetooth calling	6V2G4KSIB67TDA
		Chip, 1.69" HD Display with							via which you can stay	4I23HHMRHCCCI
		550 NITS & 70% Color Gamut,							connected to your friends,	HPOQQONRLZMH
3	B08Y1SJVV5	pTron Solero MB301 3A Micro	Computers&Accessories Acce	99.0	666.66	0.85	3.9	24870	Fast Charging & Data Sync:	AEQ2YMXSZWE01
		USB Data & Charging Cable,	ssories&Peripherals Cables&A						Solero MB301 micro USB	S56YTZQ,AGRVIN
		Made in India, 480Mbps Data	ccessories Cables USBCables						cable supports fast charge up	CWFXZYYIZOFTQ,
		Sync, Strong & Durable							to 5V/3A for devices and data	OXWGYDL57KHCN
		1.5-Meter Nylon Braided USB							syncing speed up to	AF7NDY2H6JVYTS
4	B08Y1SJVV5	pTron Solero MB301 3A Micro	Computers&Accessories Acce	99.0	666.66	0.85	3.9	24871	Fast Charging & Data Sync:	AEQ2YMXSZWEOR
		USB Data & Charging Cable,	ssories&Peripherals Cables&A						Solero MB301 micro USB	S56YTZQ,AGRVIN
		Made in India, 480Mbps Data	ccessories Cables USBCables						cable supports fast charge up	CWFXZYYIZOFTQ,
		Sync, Strong & Durable							to 5V/3A for devices and data	OXWGYDL57KHC
		1.5-Meter Nylon Braided USB							syncing speed up to	AF7NDY2H6JVYT
5	B08Y1TFSP6	pTron Solero TB301 3A	Computers&Accessories Acce	149.0	1000.0	0.85	3.9	24870	Fast Charging & Data Sync:	AEQ2YMXSZWEOR
		Type-C Data and Fast	ssories&Peripherals Cables&A						Solero TB301 Type-C cable	S56YTZQ,AGRVIN

추천 시스템 5: 같이 본 상품 기반 추천 ("함께 본 제품")

컨셉: 한 제품을 본(또는 리뷰한) 사용자들이 **같이 본/리뷰한 다른 제품들**을 집계하여, 연관 성이 높은 상품을 추천.

즉, "A를 본 사람들이 B도 봤다"는 **사용자 행동 교집합**을 이용

핵심 질문: "같은 유저들이 함께 관심 가진(또는 리뷰한) 다른 제품은 무엇일까?"

주요 컬럼: rating , rating_count

SQL 로직:

```
-- 5. 같이 본 상품 기반 추천
-- 1) (product_id, user_id) 페어 만들기
WITH exploded_reviews AS (
SELECT
  product_id,
  TRIM(user_id_each) AS user_id -- user_id의 공백 제거
 FROM 'our-highway-465502-r0.project.amazon',
   UNNEST(SPLIT(user_id, ',')) AS user_id_each
-- 2) 특정 상품을 리뷰한 user_id 목록 추출 (예: 'B07JW9H4J1')
, target_product_users AS (
 SELECT DISTINCT user_id
FROM exploded_reviews
WHERE product_id = 'B07JW9H4J1'
)
-- 3) 이 유저들이 리뷰한 다른 상품 찾기
, related_products AS (
 SELECT
  er.product_id,
  COUNT(*) AS co_review_count
 FROM exploded_reviews er
 JOIN target_product_users tpu
  ON er.user_id = tpu.user_id
 WHERE er.product_id != 'B07JW9H4J1'
 GROUP BY er.product_id
)
```

-- 4) 결과 출력 (연관도 높은 순)

SELECT *
FROM related_products
ORDER BY co_review_count DESC
LIMIT 10;

