

# Main Quest 2 템플릿

#### 평가 항목

## 추천시스템 1

- 1. 추천시스템의 이름
- 2. 추천시스템의 테마 및 목적
- 3. 구현 로직 (코드, 쿼리 및 구조도)
- 4. 결과
- 5. 참고자료

# 추천시스템 2

- 1. 추천시스템의 이름
- 2. 추천시스템의 테마 및 목적
- 3. 구현 로직 (코드, 쿼리 및 구조도)
- 4. 결과
- 5. 참고자료

#### 추천시스템 3

- 1. 추천시스템의 이름
- 2. 추천시스템의 테마 및 목적
- 3. 구현 로직 (코드, 쿼리 및 구조도)
- 4. 결과
- 5. 참고자료

## 추천시스템 4

- 1. 추천시스템의 이름
- 2. 추천시스템의 테마 및 목적
- 3. 구현 로직 (코드, 쿼리 및 구조도)
- 4. 결과
- 5. 참고자료

#### 추천시스템 5

- 1. 추천시스템의 이름
- 2. 추천시스템의 테마 및 목적
- 3. 구현 로직 (코드, 쿼리 및 구조도)
- 4. 결과
- 5. 참고자료

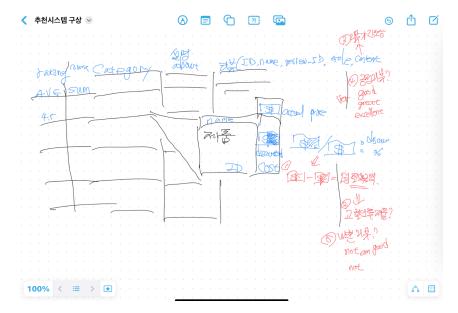
회고

# 평가 항목

채점 기준	설명
1. 추천 시스템의 창의성과 복잡성	5가지의 추천 시스템 아이디어가 모두 독창적인가?
2. SQL 구현의 정확성 및 효율성	SQL 쿼리가 데이터의 목적에 맞게 설계되었는가?
3. 보고서 구성 및 설명력	논리적으로 추천 시스템 구현 로직을 설명했는가?

## 추천시스템 1

1. 먼저 아마존 데이터 테이블 컬럼별 추천시스템 구상



## 1. 추천시스템의 이름

• 평균 할인율보다 높은 할인율의 제품들 상위 100개를 파악

## 2. 추천시스템의 테마 및 목적

• 높은 할인율의 제품들을 파악하여 향후 리뷰수와 고객의 반응을 알기위하여

## 3. 구현 로직 (코드, 쿼리 및 구조도)

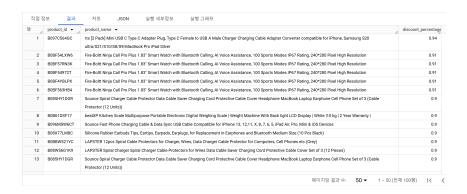
•

```
WITH AverageDiscount AS (
    SELECT AVG(discount_percentage) AS avg_discount_percentage
    FROM `modulabs-447905.amazon.ama`
)

SELECT    product_id,
    product_name,
    discount_percentage
FROM `modulabs-447905.amazon.ama`, AverageDiscount
WHERE discount_percentage > avg_discount_percentage
ORDER BY discount_percentage DESC
LIMIT 100
```

## 4. 결과

.



#### 5. 참고자료

•

## 추천시스템 2

#### 1. 추천시스템의 이름

• 고객의 만족도(긍정적인 반응)는 높은 할인율과 관계 있을까?

#### 2. 추천시스템의 테마 및 목적

• 높은 할인율이 고객의 만족도에 기여한다면 높은 할인율을 지속 확대 시키고, 고객의 만족도와 높은 할인율이 관계가 없다면 높은 할인율을 적용하는 것은 재고 하는것이 목적.

## 3. 구현 로직 (코드, 쿼리 및 구조도)

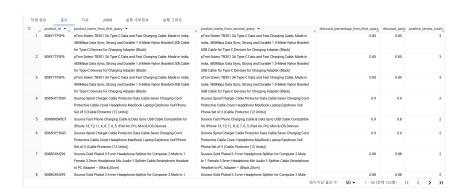
•

```
WITH AverageDiscount AS ( -- 평균 할인율 계산
 SELECT AVG(discount_percentage) AS avg_discount_percentage
  FROM `modulabs-447905.amazon.ama`
TopDiscountedProducts1 AS ( -- 첫 번째 쿼리: 평균 할인율보다 높은 상위 100개 제품
 SELECT
   product_id,
   product_name,
   discount_percentage
 FROM `modulabs-447905.amazon.ama`, AverageDiscount
 WHERE discount_percentage > avg_discount_percentage
  ORDER BY discount_percentage DESC
  LIMIT 100
),
TopDiscountedProducts2 AS ( -- 두 번째 쿼리: 긍정적인 단어가 포함된 리뷰 개수 계산 후 정렬
  WITH TopDiscountedProducts AS (
   SELECT
     product_id,
     product_name,
     discount_percentage,
     SPLIT(review_content, '|') AS split_reviews
   FROM `modulabs-447905.amazon.ama`, AverageDiscount
   WHERE discount_percentage > avg_discount_percentage
   ORDER BY discount_percentage DESC
   LIMIT 100
  )
  SELECT
   product_id,
   product_name,
   discount_percentage,
   COUNTIF(REGEXP_CONTAINS(review, r'(?i)\b(satisfied|Good|like|Good product|good experience|great dea
  FROM TopDiscountedProducts, UNNEST(split_reviews) AS review
  GROUP BY product_id, product_name, discount_percentage
  ORDER BY positive_review_count DESC
)
-- 공통된 제품 ID 찾기
SELECT
  t1.product id.
  t1.product_name AS product_name_from_first_query,
  t2.product_name AS product_name_from_second_query,
```

```
t1.discount_percentage AS discount_percentage_from_first_query,
t2.discount_percentage AS discount_percentage_from_second_query,
t2.positive_review_count
FROM TopDiscountedProducts1 t1
JOIN TopDiscountedProducts2 t2 ON t1.product_id = t2.product_id
ORDER BY t2.positive_review_count DESC;
```

## 4. 결과

.



#### 5. 참고자료

•

## 추천시스템 3

#### 1. 추천시스템의 이름

• 할인된 가격이 높은 제품들인데도 만족도나 낙관적이지 못한 제품들은 개선하자.

## 2. 추천시스템의 테마 및 목적

• 할인된 가격이 평균 할인된 가격보다 높은 제품들은 향후 수익이 줄어들면(만족도, 긍정적인 리뷰가 없는 제품들) 리스크가 크므로 개선 하는것이 목적.

## 3. 구현 로직 (코드, 쿼리 및 구조도)

•

```
WITH AverageDiscountedPrice AS ( -- 평균 discounted_price 계산
 SELECT AVG(discounted_price) AS avg_discounted_price
  FROM `modulabs-447905.amazon.ama`
),
HigherPricedProducts AS ( -- 평균 discounted_price보다 높은 제품 필터링
 SELECT
   product_id,
   product_name,
   discounted_price,
   review_content
  FROM `modulabs-447905.amazon.ama`, AverageDiscountedPrice
 WHERE discounted_price > avg_discounted_price
NonPositiveReviewProducts AS ( -- 만족도나 추천, 긍정적인 단어가 포함되지 않은 리뷰 개수 계산
  WITH TopDiscountedProducts AS (
   SELECT
     product_id,
     product_name,
```

```
SPLIT(review_content, '|') AS split_reviews
   FROM `modulabs-447905.amazon.ama`
  )
  SELECT
   product_id,
   product_name,
   COUNTIF(NOT REGEXP_CONTAINS(review,r'(?i)\b(satisfied|satisfactory|like|liked|Performance is fine|N
  FROM TopDiscountedProducts, UNNEST(split_reviews) AS review
  GROUP BY product_id, product_name
  HAVING non_positive_review_count > 0
-- 공통된 제품 찾기
SELECT
  h.product_id,
  h.product_name AS higher_priced_product_name,
  h.discounted_price,
 h.review_content, -- review_content 추가
 n.non_positive_review_count
FROM HigherPricedProducts h JOIN NonPositiveReviewProducts n ON h.product_id = n.product_id
ORDER BY h.discounted_price DESC;
```

- 1. 평균 할인된 가격보다 높은 제품을 필터링.
- 2. 제품 리뷰에서 만족도, 추천또는 권장, 긍정적인 단어가 포함되지 않은 리뷰의 개수를 계산.
- 3. 두 결과를 결합하여 공통된 제품을 반환.
- 4. 최종 결과는 할인된 가격 기준으로 내림차순 정렬.

## 4. 결과

•

```
| Special Conference | Special
```

## 5. 참고자료

.

## 추천시스템 4

## 1. 추천시스템의 이름

.

## 2. 추천시스템의 테마 및 목적

•

## 3. 구현 로직 (코드, 쿼리 및 구조도)

.

## 4. 결과

.

## 5. 참고자료

.

## 추천시스템 5

## 1. 추천시스템의 이름

•

## 2. 추천시스템의 테마 및 목적

.

## 3. 구현 로직 (코드, 쿼리 및 구조도)

•

## 4. 결과

.

## 5. 참고자료

•

## 회고

처음 데이터 컬럼들을 나열해 보면서 고민을 해 본결과, 수 많은 리뷰의 내용들 중에서 만족도 와 긍정적인 제품들과 할인된 가격의 높은 금액과의 상관 관계를 찾으려 했으나, 대부분의 제품들이 해당되어 역발상으로 만족도, 추천 또는 긍정적인 문구가 포함되지 않은 제품들로 변경하여 그중에서 할인된 금액이 높은 제품은 향후 수익에 부정적으로 적용할 것으로 예측되어 이들 제품들의 찾아서 개선하면 향후에도 발생되는 수익을 긍정적으로 볼 수 있지 않을까하여 시도 해 보았습니다.

여러 조건들을 한번에 적용할려고 할때 쉽지 않았으며, 텍스트의 조회에서 긍정과 부정을 어떻게 판단할것인가에 대한 고민으로 실제로 review\_content 의 내용들이 어떤것들이 있는지 파악하는데에도 시간이 많이 소요되었던것 같습니다. 만약 데이터의 축적시 향후 분석을 염두해 둔다면 나름의 기준을 선정하여 만족도나 불만족을 수치화 하여 사전에 컬럼으로 저장해 놓는다면 좀더 데이터를 분석하는데 수월할 수 있겠다고 느꼈습니다.