고객을 세그먼테이션하자! [프로젝트] - 이택원

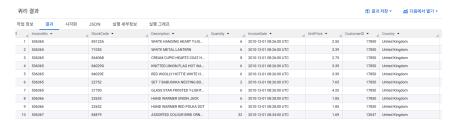
11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.data`
LIMIT 10;

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT COUNT(*) AS total_rows
FROM `smiling-mark-468901-i5.modulabs_project.data`;

[결과 이미지를 넣어주세요]



데이터 수 세기

• SELECT

COUNT(InvoiceNo) AS cnt_InvoiceNo,
COUNT(StockCode) AS cnt_StockCode,
COUNT(Description) AS cnt_Description,
COUNT(Quantity) AS cnt_Quantity,
COUNT(InvoiceDate) AS cnt_InvoiceDate,
COUNT(UnitPrice) AS cnt_UnitPrice,
COUNT(CustomerID) AS cnt_CustomerID,
COUNT(Country) AS cnt_Country
FROM `smiling-mark-468901-i5.modulabs_project.data`;



11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계산
 - 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

```
-- 각 컬럼의 누락값 비율(%) 계산 (소수점 2자리)
WITH base AS (
SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.data`
SELECT 'InvoiceNo' AS column_name,
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2) AS missing_perc
entage
FROM base
UNION ALL
SELECT 'StockCode',
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'Description',
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'Quantity',
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'InvoiceDate',
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'UnitPrice',
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
UNION ALL
SELECT 'CustomerID',
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'Country',
   ROUND(SAFE_DIVIDE(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
ORDER BY missing_percentage DESC;
```



결측치 처리 전략

• StockCode = '85123A' 의 Description 을 추출하는 쿼리문을 작성하기

SELECT DISTINCT
Description
FROM `smilling-mark-468901-i5.modulabs_project.data`
WHERE StockCode = '85123A';

[결과 이미지를 넣어주세요]



결측치 처리

• DELETE 구문을 사용하며, WHERE 절을 통해 데이터를 제거할 조건을 제시

DELETE FROM 'smiling-mark-468901-i5.modulabs_project.data' WHERE CustomerID IS NULL OR Description IS NULL;

[결과 이미지를 넣어주세요]



11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

• 중복된 행의 수를 세어보기

。 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

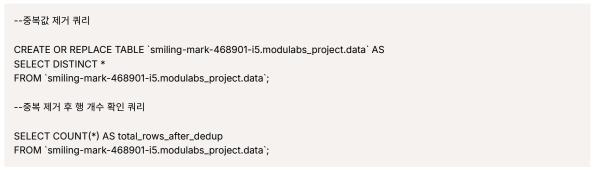
```
---- 1) 중복 그룹과 중복 행 수 집계
WITH grouped AS (
SELECT
InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, COUNT(*) AS cnt
FROM 'smiling-mark-468901-i5.modulabs_project.data'
GROUP BY 1,2,3,4,5,6,7,8
)
SELECT
-- 중복 그룹 개수(동일 조합이 2번 이상 등장한 조합의 개수)
SUM(CASE WHEN cnt > 1 THEN 1 ELSE 0 END) AS duplicate_groups,
-- 중복이 포함된 전체 행 수(해당 그룹의 행 수 합계)
SUM(CASE WHEN cnt > 1 THEN cnt ELSE 0 END) AS rows_in_duplicate_groups,
-- '중복'으로 간주되는 초과 행 수(각 그룹에서 1개를 제외한 나머지)
SUM(CASE WHEN cnt > 1 THEN cnt - 1 ELSE 0 END) AS excess_duplicate_rows
FROM grouped;
```

[결과 이미지를 넣어주세요]



중복값 처리

- 중복값을 제거하는 쿼리문 작성하기
 - CREATE OR REPLACE TABLE 구문을 활용하여 모든 컬럼(*)을 DISTINCT 한 데이터로 업데이트







11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count FROM `smiling-mark-468901-i5.modulabs_project.data`;

[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

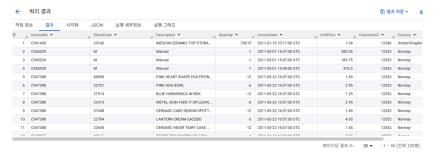
SELECT DISTINCT InvoiceNo
FROM `smiling-mark-468901-i5.modulabs_project.data`
LIMIT 100;

[결과 이미지를 넣어주세요]



• InvoiceNo 가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)

SELECT *
FROM `smilling-mark-468901-i5.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;



• 구매 건 상태가 Canceled 인 데이터의 비율(%) - 소수점 첫번째 자리까지

SELECT
ROUND(
COUNTIF(InvoiceNo LIKE 'C%') / COUNT(*) * 100,

1
) AS cancelled_ratio_percent
FROM `smilling-mark-468901-i5.modulabs_project.data`;

[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count FROM `smiling-mark-468901-i5.modulabs_project.data`;

[결과 이미지를 넣어주세요]



- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기
 - 。 상위 10개의 제품들을 출력하기

SELECT
StockCode,
COUNT(*) AS sell_cnt
FROM `smiling-mark-468901-i5.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;



작업 정	보 결과	시각화	JSON 실형	행 세부정보	실행 그래프
행 //	StockCode ▼	//	sell_cnt ▼	//	
1	85123A		20	65	
2	22423		18	94	
3	85099B		16	59	
4	47566		14	09	
5	84879		14	05	
6	20725		13	46	
7	22720		12	24	
8	POST		11	96	
9	22197		11	10	
10	23203		11	08	

- StockCode 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 。 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT
StockCode,
number_count
FROM (
SELECT
StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM 'smiling-mark-468901-i5.modulabs_project.data'
)
WHERE number_count IN (0, 1)
ORDER BY number_count, StockCode;
```

쿼리 결과



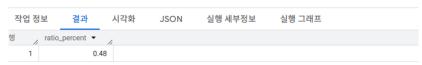
- StockCode 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 。 숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
WITH marked AS (
SELECT
StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM `smiling-mark-468901-i5.modulabs_project.data`
)
SELECT
```

ROUND(COUNTIF(number_count IN (0, 1)) / COUNT(*) * 100, 2) AS ratio_percent FROM marked;

[결과 이미지를 넣어주세요]

쿼리 결과



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `smiling-mark-468901-i5.modulabs_project.data`

WHERE StockCode IN (

SELECT DISTINCT StockCode

FROM (

SELECT

StockCode,

LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count

FROM `smiling-mark-468901-i5.modulabs_project.data`
)

WHERE number_count IN (0, 1)
);
```

[결과 이미지를 넣어주세요]

쿼리 결과



Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT
Description,
COUNT(*) AS description_cnt
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE Description IS NOT NULL
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```



• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');

[결과 이미지를 넣어주세요]



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE 'smiling-mark-468901-i5.modulabs_project.data` AS SELECT

* EXCEPT (Description),
UPPER(Description) AS Description
FROM 'smiling-mark-468901-i5.modulabs_project.data`;

[결과 이미지를 넣어주세요]



UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

SELECT

MIN(UnitPrice) AS min_price,

MAX(UnitPrice) AS max_price,

ROUND(AVG(UnitPrice), 2) AS avg_price

FROM `smiling-mark-468901-i5.modulabs_project.data`;



• 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기



[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE `smiling-mark-468901-i5.modulabs_project.data` AS SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE UnitPrice <> 0;
```

[결과 이미지를 넣어주세요]



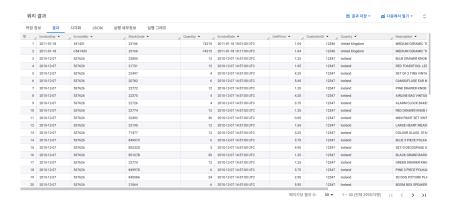
11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT
DATE(InvoiceDate) AS InvoiceDay,

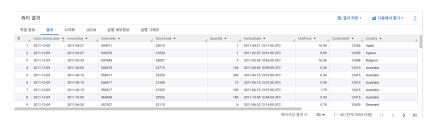
*
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT
MAX(DATE(InvoiceDate)) OVER() AS most_recent_date, -- 윈도우 함수
DATE(InvoiceDate) AS InvoiceDay,
*
FROM `smiling-mark-468901-i5.modulabs_project.data`;

[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY InvoiceDay DESC;

[결과 이미지를 넣어주세요]



• 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산하기

```
SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM smilling-mark-468901-i5.modulabs_project.data
GROUP BY CustomerID
);
```



• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user_r 이라는 이름의 테이블로 저장하기

```
CREATE OR REPLACE TABLE `smiling-mark-468901-i5.modulabs_project.user_r` AS
WITH base AS (
SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.data`
 WHERE CustomerID IS NOT NULL
  AND Description IS NOT NULL
  AND UnitPrice <> 0
  AND (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) NOT IN (0, 1)
),
last_purchase AS (
SELECT
 CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
 FROM base
 GROUP BY CustomerID
),
most_recent AS (
SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
FROM base
)
SELECT
lp.CustomerID,
 lp.InvoiceDay AS last_purchase_date,
 mr.most_recent_date,
DATE_DIFF(mr.most_recent_date, lp.InvoiceDay, DAY) AS recency_days
FROM last_purchase lp
CROSS JOIN most_recent mr;
```



Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY purchase_cnt DESC;

[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

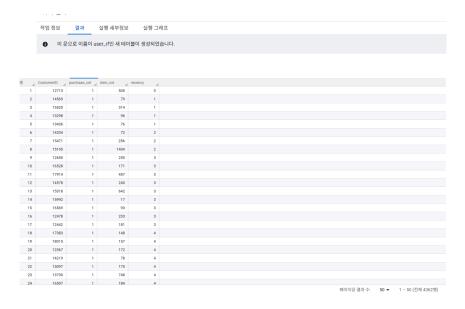
SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID;



• 전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 user_rf 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE `smiling-mark-468901-i5.modulabs_project.user_rf` AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
 CustomerID,
 COUNT(DISTINCT InvoiceNo) AS purchase_cnt
 FROM 'smiling-mark-468901-i5.modulabs_project.data'
 WHERE CustomerID IS NOT NULL
 GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
 SUM(Quantity) AS item_cnt
 FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
)
-- user_r와 결합
SELECT
pc.CustomerID,
pc.purchase_cnt,
ic.item_cnt,
ur.recency_days AS recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
ON pc.CustomerID = ic.CustomerID
JOIN `smiling-mark-468901-i5.modulabs_project.user_r` AS ur
ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]



Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
AND Quantity > 0 -- 반품/취소(음수 수량) 제외
GROUP BY CustomerID
ORDER BY user_total DESC;
```

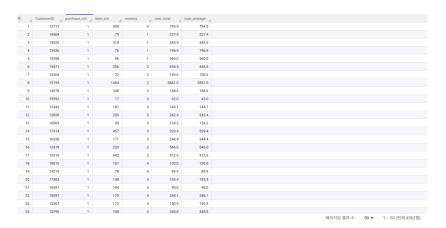


• 고객별 평균 거래 금액 계산

○ 고객별 평균 거래 금액을 구하기 위해 1) data 테이블을 user_rf 테이블과 조인(LEFT JOIN) 한 후, 2) purchase_cnt 로 나누어서 3) user_rfm 테이블로 저장하기

```
CREATE OR REPLACE TABLE 'smiling-mark-468901-i5.modulabs_project.user_rfm' AS
SELECT
 rf.CustomerID AS CustomerID,
 rf.purchase_cnt,
 rf.item_cnt,
 rf.recency,
 ROUND(ut.user_total, 1) AS user_total,
 ROUND(SAFE_DIVIDE(ut.user_total, rf.purchase_cnt), 1) AS user_average
FROM `smiling-mark-468901-i5.modulabs_project.user_rf` rf
LEFT JOIN (
 -- 고객 별 총 지출액
 SELECT
  CustomerID,
  SUM(CASE WHEN Quantity > 0 THEN Quantity * UnitPrice ELSE 0 END) AS user_total
 FROM `smiling-mark-468901-i5.modulabs_project.data`
 WHERE CustomerID IS NOT NULL
 GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```





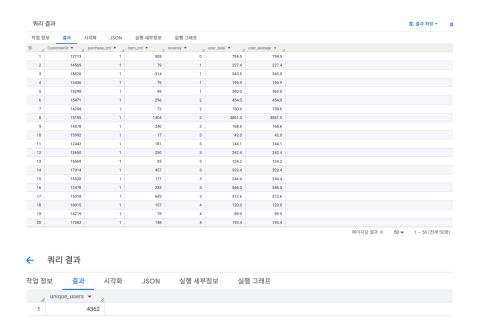
RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.user_rfm`
LIMIT 50;

--고유한 유저 수는 user_rfm이 고객 1명당 1행이므로 COUNT
SELECT COUNT(*) AS unique_users
FROM `smiling-mark-468901-i5.modulabs_project.user_rfm`;
```

[결과 이미지를 넣어주세요]



11-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

• 1) 고객 별로 구매한 상품들의 고유한 수를 계산하기

2) user_rfm 테이블과 결과를 합치기

3) user_data 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE smiling-mark-468901-i5.modulabs_project.user_data AS

WITH unique_products AS (

SELECT

CustomerID,

COUNT(DISTINCT StockCode) AS unique_products

FROM smiling-mark-468901-i5.modulabs_project.data

GROUP BY CustomerID
)

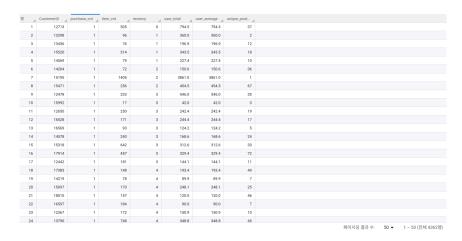
SELECT ur.*, up.* EXCEPT (CustomerID)

FROM smiling-mark-468901-i5.modulabs_project.user_rfm AS ur

JOIN unique_products AS up

ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

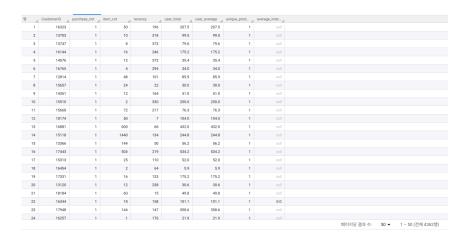


2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 균 구매 소요 일수를 계산하고, 그 결과를 user_data 에 통합

```
CREATE OR REPLACE TABLE 'smiling-mark-468901-i5.modulabs_project.user_data' AS
WITH purchase_intervals AS (
-- (2) 고객별 평균 구매 간격(일)
SELECT
 CustomerID,
 COALESCE(ROUND(AVG(interval_), 2), 0) AS average_interval
 -- (1) 연속 구매 간 일수
 SELECT
  CustomerID,
  DATE_DIFF(
   DATE(InvoiceDate),
   LAG(DATE(InvoiceDate)) OVER (PARTITION BY CustomerID ORDER BY DATE(InvoiceDate)),
   DAY
  ) AS interval_
  FROM 'smiling-mark-468901-i5.modulabs_project.data'
 WHERE CustomerID IS NOT NULL
WHERE interval_ IS NOT NULL
GROUP BY CustomerID
```

```
SELECT
u.* EXCEPT (average_interval), -- u의 average_interval 제거
pi.average_interval -- pi의 값을 최종 컬럼으로 사용
FROM `smiling-mark-468901-i5.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```



3. 구매 취소 경향성

• 고객의 취소 패턴 파악하기

(취소 비율은 소수점 두번째 자리)

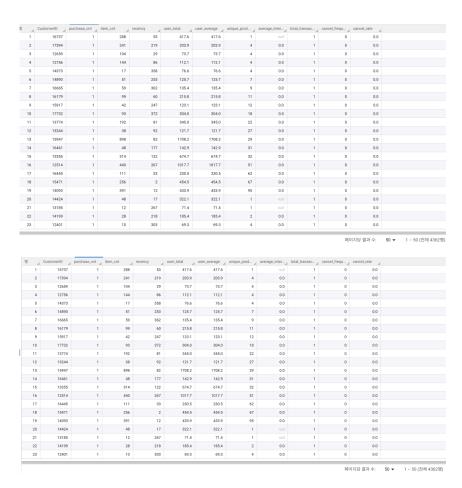
1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수 2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율

○ 취소 빈도와 취소 비율을 계산하고 그 결과를 user_data 에 통합하기

CREATE OR REPLACE TABLE 'smiling-mark-468901-i5.modulabs_project.user_data' AS

```
WITH TransactionInfo AS (
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS total_transactions,
COUNT(DISTINCT IF(InvoiceNo LIKE 'C%', InvoiceNo, NULL)) AS cancel_frequency
FROM 'smiling-mark-468901-i5.modulabs_project.data'
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
)

SELECT
u.*,
t.* EXCEPT (CustomerID),
ROUND(SAFE_DIVIDE(t.cancel_frequency, t.total_transactions) * 100, 2) AS cancel_rate
FROM 'smiling-mark-468901-i5.modulabs_project.user_data' AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

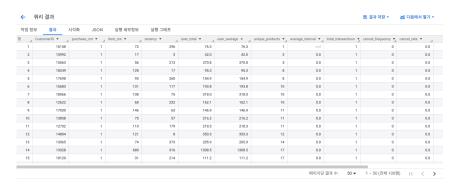


• 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 user_data 를 출력하기

SELECT *

FROM `smiling-mark-468901-i5.modulabs_project.user_data` LIMIT 100;

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

Keep:

Problem :

Try: