

고객을 세그멘테이션하자! [프로젝트] - 김채린

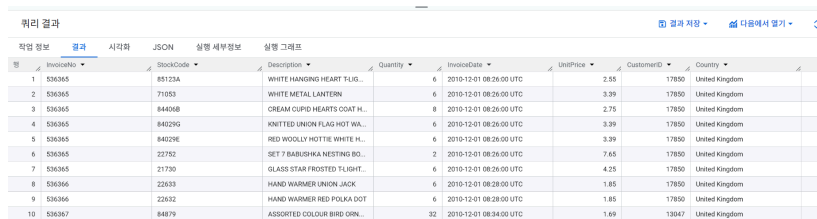
11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM `constant-abacus-470206-f2.modulabs_project.data`  
LIMIT 10;
```

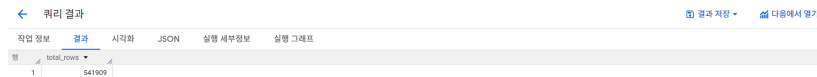
[결과 이미지를 넣어주세요]



- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*) AS total_rows  
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



데이터 수 세기

- COUNT 함수를 사용해서, 각 컬럼별 데이터 포인트의 수를 세어 보기

```
SELECT  
COUNT(*) AS total_rows,  
COUNT(InvoiceNo) AS count_InvoiceNo,  
COUNT(StockCode) AS count_StockCode,  
COUNT(Description) AS count_Description,  
COUNT(Quantity) AS count_Quantity,  
COUNT(InvoiceDate) AS count_InvoiceDate,  
COUNT(UnitPrice) AS count_UnitPrice,  
COUNT(CustomerID) AS count_CustomerID,  
COUNT(Country) AS count_Country  
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

결과 저장

다음에서 보기

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

항	total_rows	count_InvoiceNo	count_StockCode	count_Description	count_Quantity	count_InvoiceDate	count_UnitPrice	count_CustomerID	count_Country
1	541909	541909	541909	540455	541909	541909	541909	406829	541909

11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계
 - 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 **UNION ALL**을 통해 합치기

```

SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `constant-abacus-470206-f2.modulabs_project.data`

UNION ALL
SELECT
  'StockCode',
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `constant-abacus-470206-f2.modulabs_project.data`

UNION ALL
SELECT
  'Description',
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `constant-abacus-470206-f2.modulabs_project.data`

UNION ALL
SELECT
  'Quantity',
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `constant-abacus-470206-f2.modulabs_project.data`

UNION ALL
SELECT
  'InvoiceDate',
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `constant-abacus-470206-f2.modulabs_project.data`

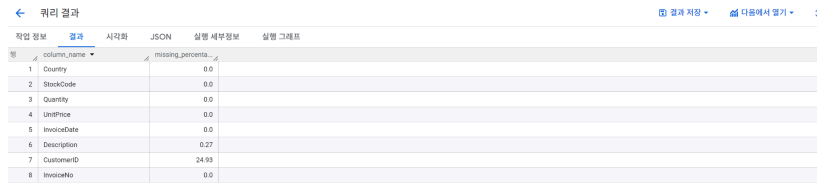
UNION ALL
SELECT
  'UnitPrice',
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `constant-abacus-470206-f2.modulabs_project.data`

UNION ALL
SELECT
  'CustomerID',
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `constant-abacus-470206-f2.modulabs_project.data`

UNION ALL
SELECT
  'Country',
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `constant-abacus-470206-f2.modulabs_project.data`;

```

[결과 이미지를 넣어주세요]



row	column_name	missing_percent
1	Country	0.0
2	StockCode	0.0
3	Quantity	0.0
4	UnitPrice	0.0
5	InvoiceDate	0.0
6	Description	0.27
7	CustomerID	24.93
8	InvoiceNo	0.0

결측치 처리 전

- **StockCode = '85123A' 의 Description** 을 추출하는 쿼리문을 작성하기

```
SELECT DISTINCT Description
FROM `constant-abacus-470206-f2.modulabs_project.data`
WHERE TRIM(StockCode) = '85123A';
```

[결과 이미지를 넣어주세요]



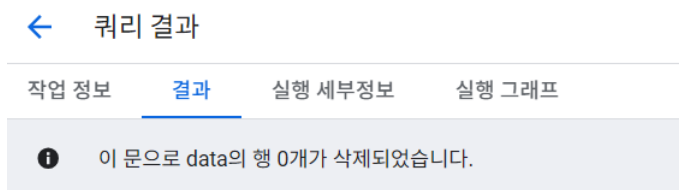
row	Description
1	WHITE HANGING HEART T-LIG...
2	CREAM HANGING HEART T-LIG...

결측치 처리

- **DELETE** 구문을 사용하며, **WHERE** 절을 통해 데이터를 제거할 조건을 제시

```
DELETE
FROM `constant-abacus-470206-f2.modulabs_project.data`
WHERE CustomerID IS NULL;
```

[결과 이미지를 넣어주세요]



작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 data의 행 0개가 삭제되었습니다.			

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- 중복된 행의 수를 세어보기
 - 8개의 컬럼에 그룹 함수를 적용한 후, **COUNT**가 1보다 큰 데이터를 세어보기

```
SELECT COUNT(*) AS duplicate_groups
FROM (
```

```
SELECT
  InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country,
  COUNT(*) AS cnt
FROM `constant-abacus-470206-f2.modulabs_project.data`
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING COUNT(*) > 1
);
```

[결과 이미지를 넣어주세요]

← 쿼리 결과	
작업 정보	결과
시각화	JSON
실행 세부정보	실행 그래프
행	duplicate_groups
1	4837

중복값 처리

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

```
CREATE OR REPLACE TABLE `constant-abacus-470206-f2.modulabs_project.data` AS
SELECT DISTINCT *
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과	
작업 정보	결과
실행 세부정보	실행 그래프
<p>i 이 문으로 이름이 data인 테이블이 교체되었습니다.</p>	

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	unique_invoice_c...				
1	22190				

- 고유한 **InvoiceNo** 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM `constant-abacus-470206-f2.modulabs_project.data`
ORDER BY InvoiceNo
LIMIT 100;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	시각화	JSON
행	InvoiceNo		
1	536365		
2	536366		
3	536367		
4	536368		
5	536369		
6	536370		
7	536371		
8	536372		
9	536373		
10	536374		
11	536375		
12	536376		
13	536377		

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

```
SELECT *
FROM `constant-abacus-470206-f2.modulabs_project.data`
WHERE CAST(InvoiceNo AS STRING) LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다들여서 보기

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프			
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
6	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
7	C547388	84050	PINK-HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
8	C547388	22701	PINK GOO BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway

페이지당 결과 수: 50

1 ~ 50 (전체 100행)

<

>

>

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

```
SELECT
  ROUND(
    SUM(CASE WHEN CAST(InvoiceNo AS STRING) LIKE 'C%' THEN 1 ELSE 0 END)
    / COUNT(*) * 100, 1
  ) AS canceled_ratio_pct
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	canceled_ratio_pct				
1	2.2				

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	unique_stockcod...				
1	3684				

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

```
SELECT
  StockCode,
  COUNT(*) AS sell_cnt
FROM `constant-abacus-470206-f2.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	StockCode ▼	sell_cnt ▼			
1	85123A	2065			
2	22423	1894			
3	85099B	1659			
4	47566	1409			
5	84879	1405			
6	20725	1346			
7	22720	1224			
8	POST	1196			
9	22197	1110			
10	23203	1108			

- **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 `constant-abacus-470206-f2.modulabs_project.data`
)
WHERE number_count BETWEEN 0 AND 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	StockCode ▼	number_count ▼			
1	POST	0			
2	M	0			
3	C2	1			
4	D	0			
5	BANK CHARGES	0			
6	PADS	0			
7	DOT	0			
8	CRUK	0			

- **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)

```
SELECT
  ROUND(
    SUM(CASE WHEN number_count BETWEEN 0 AND 1 THEN 1 ELSE 0 END) / COUNT(*) * 100,
```

```

2
) AS ratio_pct
FROM (
SELECT
  StockCode,
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM `project_name.modulabs_project.data`
);

```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보	결과	시각화
JSON	실행 세부정보	실행 그래프
행	ratio_pct	
1	0.48	

- 제품과 관련되지 않은 거래 기록을 제거하기

```

DELETE FROM `constant-abacus-470206-f2.modulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM `constant-abacus-470206-f2.modulabs_project.data`
  WHERE StockCode IN ('POST', 'D', 'M', 'BANK CHARGES', 'ADJUST', 'AMAZONFEE', 'C2')
);

```

[결과 이미지를 넣어주세요]


쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 data의 행 1,879개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```

SELECT
  Description,
  COUNT(*) AS description_cnt
FROM `constant-abacus-470206-f2.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;

```



[결과 이미지를 넣어주세요]

쿼리 결과			
작업 정보		결과	시각화
		JSON	실행 세부정보
		실행 세부정보	실행 그래프
행	Description ▾	description_cnt ▾	
1	WHITE HANGING HEART T-LIG...	2058	
2	REGENCY CAKESTAND 3 TIER	1894	
3	JUMBO BAG RED RETROSPOT	1659	
4	PARTY BUNTING	1409	
5	ASSORTED COLOUR BIRD ORN...	1405	
6	LUNCH BAG RED RETROSPOT	1345	
7	SET OF 3 CAKE TINS PANTRY D...	1224	
8	LUNCH BAG BLACK SKULL.	1099	
9	PACK OF 72 RETROSPOT CAKE ...	1062	
10	SPOTTY BUNTING	1026	
11	PAPER CHAIN KIT 50'S CHRIST...	1013	
12	LUNCH BAG SPACEBOY DESIGN	1006	
13	LUNCH BAG CARS BLUE	1000	
14	HEART OF WICKER SMALL	990	
15	NATURAL SLATE HEART CHAL...	989	
16	JAM MAKING SET WITH JARS	966	

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM `constant-abacus-470206-f2.modulabs_project.data`
WHERE UPPER(Description) LIKE '%POSTAGE%'
OR UPPER(Description) LIKE '%CARRIAGE%'
OR UPPER(Description) LIKE '%BANK CHARGE%'
OR UPPER(Description) LIKE '%ADJUST%'
OR UPPER(Description) LIKE '%CHECK%'
OR UPPER(Description) LIKE '%AMAZON%';
```

[결과 이미지를 넣어주세요]

쿼리 결과			
작업 정보		결과	실행 세부정보
		실행 세부정보	실행 그래프
 이 문으로 data의 행 317개가 삭제되었습니다.			

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `constant-abacus-470206-f2.modulabs_project.data` AS
SELECT
* EXCEPT (Description),
```

```
UPPER(Description) AS Description
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM `constant-abacus-470206-f2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	min_price	max_price	avg_price		
1	0.0	1100.44	2.923289500961...		

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

```
SELECT
  COUNT(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  ROUND(AVG(Quantity), 2) AS avg_quantity
FROM `constant-abacus-470206-f2.modulabs_project.data`
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	cnt_quantity	min_quantity	max_quantity	avg_quantity	
1	34	1	12540	408.18	

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT *
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT # [[YOUR QUERY]] AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
# [[YOUR QUERY]] AS most_recent_date,
# [[YOUR QUERY]] AS InvoiceDay,
*
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
# [[YOUR QUERY]] AS InvoiceDay
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];
```

[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 project_name.modulabs_project.data
  GROUP BY CustomerID
);

```

[결과 이미지를 넣어주세요]

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

```

CREATE OR REPLACE TABLE project_name.modulabs_project.user_r AS
# [[YOUR QUERY]]

```

[결과 이미지를 넣어주세요]

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```

SELECT
  CustomerID,
  # [[YOUR QUERY]] AS purchase_cnt
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];

```

[결과 이미지를 넣어주세요]

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```

SELECT
  CustomerID,
  # [[YOUR QUERY]] AS item_cnt
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];

```

[결과 이미지를 넣어주세요]

- 전체 거래 건수 계산과 구매한 아이템의 총 수량 계산의 결과를 합쳐서 `user_rf` 라는 이름의 테이블에 저장하기

```

CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  # [[YOUR QUERY]]
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  # [[YOUR QUERY]]
)

```

```
-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN project_name.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,
  # [[YOUR QUERY]] AS user_total
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

- 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 project_name.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  # [[YOUR QUERY]] AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    # [[YOUR QUERY]]
  ) ut
  ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

RFM 통합 테이블 출력하기

- 최종 `user_rfm` 테이블을 출력하기

```
# [[YOUR QUERY]];
```

[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 project_name.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM project_name.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)
 - 군 구매 소요 일수를 계산하고, 그 결과를 `user_data` 에 통합

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS
WITH purchase_intervals AS (
  -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
    FROM
      project_name.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기

- 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
(취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    # [[YOUR QUERY]] AS total_transactions,
    # [[YOUR QUERY]] AS cancel_frequency
  FROM project_name.modulabs_project.data
  # [[YOUR QUERY]]
)

SELECT u.*, t.* EXCEPT(CustomerID), # [[YOUR QUERY]] AS cancel_rate
FROM `project_name.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

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

```
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

회고

[회고 내용을 작성해주세요]

Keep :

Problem :

Try :