고객을 세그먼테이션하자! [프로젝트] - 윤선정

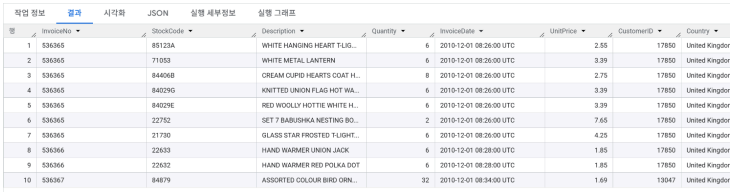
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM axiomatic-robot-470207-s8.modulabs_project.data
LIMIT 10
```

[결과 이미지를 넣어주세요]

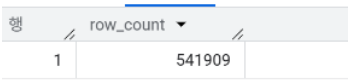


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdor
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdor
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdor
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdor
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdor
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdor
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdor
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdor
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdor
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdor

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*) AS row_count
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



행	row_count
1	541909

*참고

컬럼명	설명
InvoiceNo	각각의 고유한 거래를 나타내는 코드. 이 코드가 'C'라는 글자로 시작한다면, 취소를 나타냅니다. 하나의 거래에 여러 개의 제품이 함께 구매되었다면, 1개의 InvoiceNo에는 여러 개의 StockCode가 연결되어 있습니다.
StockCode	각각의 제품에 할당된 고유 코드
Description	각 제품에 대한 설명
Quantity	거래에서 제품을 몇 개 구매했는지에 대한 단위 수
InvoiceDate	거래가 일어난 날짜와 시간
UnitPrice	제품 당 단위 가격(영국 파운드)
CustomerID	각 고객에게 할당된 고유 식별자 코드
Country	주문이 발생한 국가

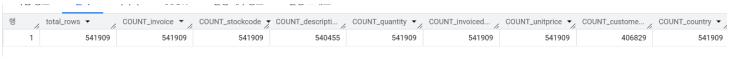
데이터 수 세기

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

```
SELECT
  COUNT(*) AS total_rows,
  COUNT(InvoiceNo)    AS COUNT_invoice,
  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 `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



행	total_rows	COUNT_invoice	COUNT_stockcode	COUNT_descripti...	COUNT_quantity	COUNT_invoiced...	COUNT_unitprice	COUNT_custome...	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 계산**
  - **각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 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 `axiomatic-robot-470207-s8.modulabs_project.data`

UNION ALL

SELECT
   'StockCode' AS column_name,
   ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `axiomatic-robot-470207-s8.modulabs_project.data`

UNION ALL

SELECT
   'Description' AS column_name,
   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `axiomatic-robot-470207-s8.modulabs_project.data`

UNION ALL

SELECT
   'Quantity' AS column_name,
   ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `axiomatic-robot-470207-s8.modulabs_project.data`

UNION ALL

SELECT
   'InvoiceDate' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `axiomatic-robot-470207-s8.modulabs_project.data`

UNION ALL

SELECT
   'UnitPrice' AS column_name,
   ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `axiomatic-robot-470207-s8.modulabs_project.data`

UNION ALL

SELECT
   'CustomerID' AS column_name,
   ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
```

```
UNION ALL

SELECT
    'Country' AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

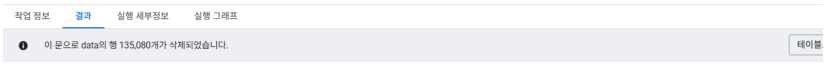
행	Description
1	WHITE HANGING HEART T-LIG...
2	?
3	wrongly marked carton 22804
4	CREAM HANGING HEART T-LIG...

결측치 처리

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

```
DELETE FROM `axiomatic-robot-470207-s8.modulabs_project.data`
WHERE CustomerID IS NULL
   OR Description IS NULL;
```

[결과 이미지를 넣어주세요]



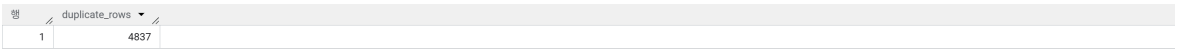
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT COUNT(*) AS duplicate_rows
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS row_count
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  HAVING COUNT(*) > 1   -- 중복된 경우만 남김
);
```

[결과 이미지를 넣어주세요]



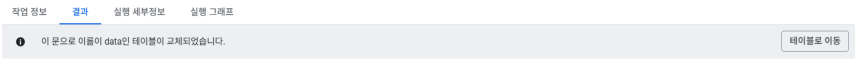
행	duplicate_rows
1	4837

중복값 처리

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

```
CREATE OR REPLACE TABLE `axiomatic-robot-470207-s8.modulabs_project.data` AS
SELECT DISTINCT *
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



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 `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	unique_invoice_c...
1	22190

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

```
SELECT DISTINCT InvoiceNo
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
ORDER BY InvoiceNo
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo
18	536384
19	536385
20	536386
21	536387
22	536388
23	536389

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

```
SELECT *
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
45	C549253	22466	FAIRY TALE COTTAGE NIGHTLI...	-1	2011-04-07 12:20:00 UTC	1.95	12408	Belgium
46	C549253	22727	ALARM CLOCK BAKELIKE RED	-1	2011-04-07 12:20:00 UTC	3.75	12408	Belgium
47	C549253	20712	JUMBO BAG WOODLAND ANIM...	-1	2011-04-07 12:20:00 UTC	2.08	12408	Belgium
48	C549253	22328	ROUND SNACK BOXES SET OF ...	-1	2011-04-07 12:20:00 UTC	2.95	12408	Belgium
49	C549253	22725	ALARM CLOCK BAKELIKE CHO...	-1	2011-04-07 12:20:00 UTC	3.75	12408	Belgium
50	C549533	22628	PICNIC BOXES SET OF 3 RETRO...	-3	2011-04-08 17:08:00 UTC	4.95	12408	Belgium

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

```
SELECT
  ROUND(
    SAFE_DIVIDE(
      SUM(CASE WHEN STARTS_WITH(InvoiceNo, 'C') THEN 1 ELSE 0 END),
      COUNT(*)
    ) * 100
  , 1) AS canceled_row_pct
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	canceled_row_pct
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

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 `axiomatic-robot-470207-s8.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

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 값**들에는 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 `axiomatic-robot-470207-s8.modulabs_project.data`
)
WHERE number_count IN (0, 1);
```

[결과 이미지를 넣어주세요]

행	StockCode	number_count
3	C2	1
4	D	0
5	BANK CHARGES	0
6	PADS	0
7	DOT	0
8	CRUK	0

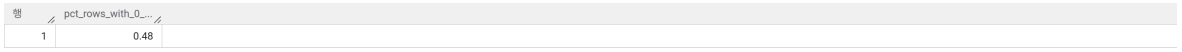
- **`StockCode`의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
-- 목록
SELECT DISTINCT StockCode, number_count, letter_count
FROM (
```

```
  SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
    LENGTH(REGEXP_REPLACE(StockCode, r'[^A-Za-z]', '')) AS letter_count
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
)
WHERE number_count IN (0, 1);

-- 비율
SELECT
  ROUND(
    SAFE_DIVIDE(
      SUM(CASE WHEN number_count IN (0, 1) THEN 1 ELSE 0 END),
      COUNT(*)
    ) * 100
  , 2) AS pct_rows_with_0_1_digits
FROM (
  SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
);
```

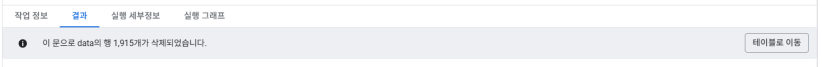
[결과 이미지를 넣어주세요]



- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `axiomatic-robot-470207-s8.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 `axiomatic-robot-470207-s8.modulabs_project.data`
  )
  WHERE number_count IN (0, 1)
);
```

[결과 이미지를 넣어주세요]



Description 살펴보기

- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS description_cnt
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

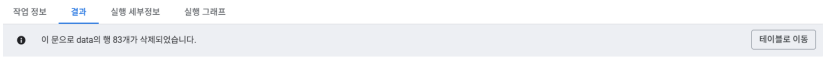
[결과 이미지를 넣어주세요]

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

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
WHERE UPPER(Description) IN ('NEXT DAY CARRIAGE', 'HIGH RESOLUTION IMAGE');
```

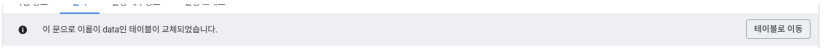
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `axiomatic-robot-470207-s8.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  ROUND(AVG(UnitPrice), 2) AS avg_price
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	min_price	max_price	avg_price
1	0.0	649.5	2.9

- **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 `axiomatic-robot-470207-s8.modulabs_project.data`
WHERE UnitPrice = 0;
```

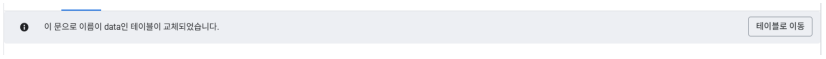
[결과 이미지를 넣어주세요]

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.52

- `UnitPrice = 0` **를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE `axiomatic-robot-470207-s8.modulabs_project.data` AS
SELECT *
FROM `axiomatic-robot-470207-s8.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 `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(DATE(InvoiceDate)) OVER() AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `axiomatic-robot-470207-s8.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
49	2011-12-09	2011-05-20	554016	21056	2	2011-05-20 13:14:00 UTC	8.95	13243	United King
50	2011-12-09	2011-03-17	546850	16238	28	2011-03-17 13:20:00 UTC	0.21	13268	United King

- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07

- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

```
SELECT
  CustomerID,
  InvoiceDay,
  EXTRACT(DAY FROM (MAX(InvoiceDay) OVER () - InvoiceDay)) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
  GROUP BY CustomerID
);
```

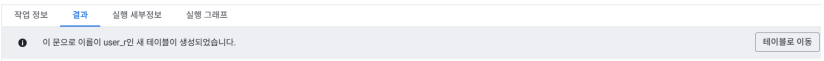
[결과 이미지를 넣어주세요]

행	CustomerID	InvoiceDay	recency
1	12622	2011-04-21	232
2	12703	2011-10-06	64

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

```
CREATE OR REPLACE TABLE `axiomatic-robot-470207-s8.modulabs_project.user_r` AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM (MAX(InvoiceDay) OVER () - InvoiceDay)) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
38	12393	4
39	12394	2

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `axiomatic-robot-470207-s8.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	item_cnt
1	12346	0
2	12347	2458

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

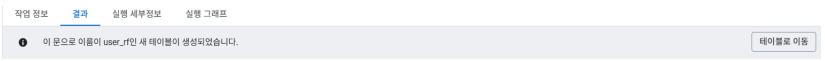
```
CREATE OR REPLACE TABLE `axiomatic-robot-470207-s8.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
  GROUP BY CustomerID
)

-- 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 `axiomatic-robot-470207-s8.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 `axiomatic-robot-470207-s8.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

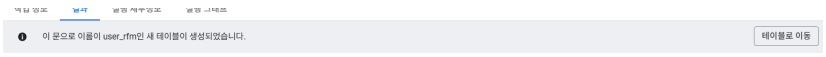
행	CustomerID	user_total
22	12371	1528.0
23	12372	1196.0

- **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 `axiomatic-robot-470207-s8.modulabs_project.user_rfm` AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(SAFE_DIVIDE(ut.user_total, rf.purchase_cnt), 1) AS user_average
FROM `axiomatic-robot-470207-s8.modulabs_project.user_rf` rf
LEFT JOIN (

  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM `axiomatic-robot-470207-s8.modulabs_project.user_rfm`
ORDER BY CustomerID
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12346	2	0	325	0.0	0.0
2	12347	7	2458	2	4310.0	615.7

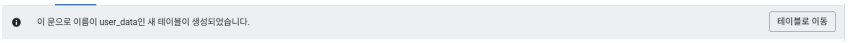
11-8. 추가 Feature 추출

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 `axiomatic-robot-470207-s8.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `axiomatic-robot-470207-s8.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT
  ur.*,
  up.unique_products
FROM `axiomatic-robot-470207-s8.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
  ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...
1	15389	1	400	172	500.0	500.0	1
2	14705	1	100	198	179.0	179.0	1
3	16738	1	3	297	3.8	3.8	1
4	17331	1	16	123	175.2	175.2	1
5	15488	1	72	92	76.3	76.3	1
6	16093	1	20	106	17.0	17.0	1
7	16428	1	-1	81	-3.0	-3.0	1
8	12791	1	96	373	177.6	177.6	1
9	16454	1	2	64	5.9	5.9	1
10	13829	1	-12	359	-102.0	-102.0	1
11	17986	1	10	56	20.8	20.8	1
12	16148	1	72	296	76.3	76.3	1
13	16138	1	-1	368	-8.0	-8.0	1
14	17347	1	216	86	229.0	229.0	1
15	15313	1	25	110	52.0	52.0	1
16	13302	1	5	155	63.8	63.8	1
17	17948	1	144	147	358.6	358.6	1
18	15316	1	100	326	165.0	165.0	1
19	13391	1	4	203	59.8	59.8	1
20	17763	1	12	263	15.0	15.0	1

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 `axiomatic-robot-470207-s8.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(
        DATE(InvoiceDate),
        LAG(DATE(InvoiceDate)) OVER (PARTITION BY CustomerID ORDER BY DATE(InvoiceDate)),
        DAY
      ) AS interval_
    FROM
      `axiomatic-robot-470207-s8.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
```

```
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `axiomatic-robot-470207-s8.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	17948	1	144	147	358.6	358.6	1	0.0
2	13307	1	4	120	15.0	15.0	1	0.0
3	16093	1	20	106	17.0	17.0	1	0.0
4	17347	1	216	86	229.0	229.0	1	0.0
5	18133	1	1350	212	931.5	931.5	1	0.0
6	16323	1	50	196	207.5	207.5	1	0.0
7	14090	1	72	324	76.3	76.3	1	0.0
8	13366	1	144	50	56.2	56.2	1	0.0

3. 구매 취소 경향성

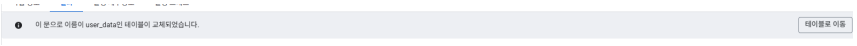
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE `axiomatic-robot-470207-s8.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT IF(Quantity < 0, InvoiceNo, NULL)) AS cancel_frequency
  FROM `axiomatic-robot-470207-s8.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  -- 취소 비율(%)
FROM `axiomatic-robot-470207-s8.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
  ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...	total_transac...	cancel_frequ...	cancel_rate
1	17102	1	2	261	25.5	25.5	1	0.0	1	0	0.0
2	13011	1	13	372	50.6	50.6	3	0.0	1	0	0.0
3	17640	1	250	86	621.7	621.7	5	0.0	1	0	0.0
4	17556	1	101	205	157.9	157.9	6	0.0	1	0	0.0
5	16430	1	103	57	300.9	300.9	6	0.0	1	0	0.0
6	15775	1	68	245	104.9	104.9	7	0.0	1	0	0.0
7	13866	1	57	74	145.7	145.7	7	0.0	1	0	0.0
8	14641	1	90	21	110.5	110.5	7	0.0	1	0	0.0
9	16315	1	75	330	226.4	226.4	14	0.0	1	0	0.0
10	17789	1	74	281	225.9	225.9	14	0.0	1	0	0.0
11	12532	1	151	30	313.8	313.8	15	0.0	1	0	0.0
12	15127	1	251	65	406.5	406.5	16	0.0	1	0	0.0
13	14881	1	316	19	255.4	255.4	17	0.0	1	0	0.0
14	18105	1	46	37	113.5	113.5	18	0.0	1	0	0.0
15	16751	1	1031	30	1764.7	1764.7	24	0.0	1	0	0.0
16	16720	1	180	26	152.9	152.9	25	0.0	1	0	0.0
17	13721	1	316	36	524.1	524.1	28	0.0	1	0	0.0
18	12475	1	532	53	669.4	669.4	35	0.0	1	0	0.0
19	12713	1	505	0	794.5	794.5	37	0.0	1	0	0.0
20	12638	1	573	33	511.7	511.7	38	0.0	1	0	0.0
21	14076	1	79	129	118.7	118.7	43	0.0	1	0	0.0
22	16274	1	151	373	331.5	331.5	63	0.0	1	0	0.0
23	16800	1	524	11	1162.7	1162.7	148	0.0	1	0	0.0
24	16881	1	600	66	432.0	432.0	1	0.0	1	0	0.0
25	16754	1	4280	372	2002.4	2002.4	2	0.0	1	0	0.0
26	14080	1	48	32	45.6	45.6	4	0.0	1	0	0.0

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

```
SELECT *
FROM `axiomatic-robot-470207-s8.modulabs_project.user_data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	17102	1	2	261	25.5	25.5	1	0.0	1	0	0.0
2	13011	1	13	372	50.6	50.6	3	0.0	1	0	0.0
3	17640	1	250	86	621.7	621.7	5	0.0	1	0	0.0
4	17556	1	101	205	157.9	157.9	6	0.0	1	0	0.0
5	16430	1	103	57	300.9	300.9	6	0.0	1	0	0.0
6	15775	1	68	245	104.9	104.9	7	0.0	1	0	0.0
7	13866	1	57	74	145.7	145.7	7	0.0	1	0	0.0
8	14641	1	90	21	110.5	110.5	7	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep :

시작전 전체적인 흐름을 한 번 생각하고 진행

끝난 후 GPT의 도움을 많이 받아서, 한 번 스스로 생각해보는 복습 작업

Problem :

많은 부분을 GPT에 의지함 (많은 코드가 생각이 잘 나지 않음)

Try :

전체적인 내용을 머리로 생각해 보는것 뿐 아니라 손으로 써보기