

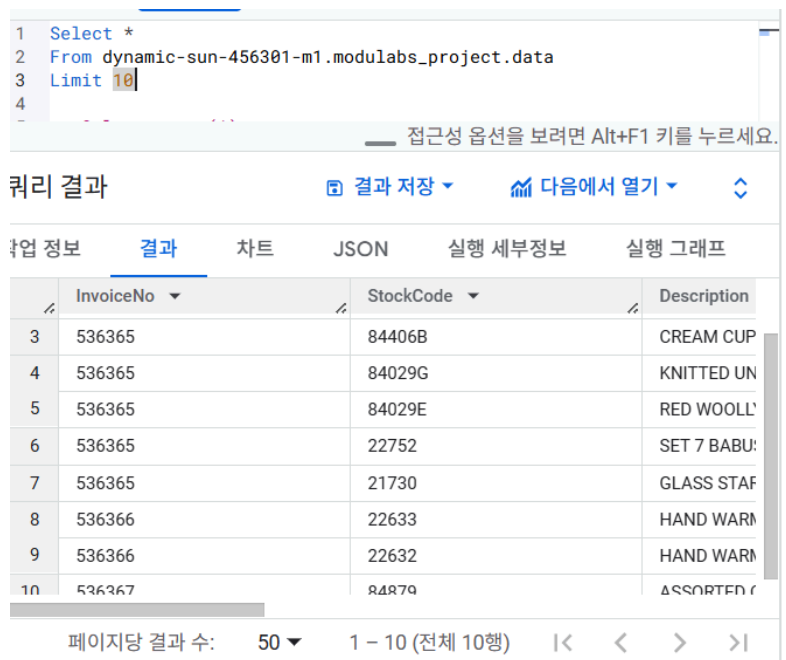
고객 세그멘테이션 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
Select *  
From dynamic-sun-456301-m1.modulabs_project.data  
Limit 10
```



The screenshot shows a SQL query execution interface. The query is: `Select *
From dynamic-sun-456301-m1.modulabs_project.data
Limit 10`. Below the query, there are tabs for '쿼리 결과' (Query Results), '결과 저장' (Save Results), '다음에서 열기' (Open in Next), and '실행 정보' (Execution Info). The '쿼리 결과' tab is selected, showing a table with 10 rows. The table has columns: InvoiceNo, StockCode, and Description. The data is as follows:

	InvoiceNo	StockCode	Description
3	536365	84406B	CREAM CUP
4	536365	84029G	KNITTED UN
5	536365	84029E	RED WOOLL
6	536365	22752	SET 7 BABU
7	536365	21730	GLASS STAF
8	536366	22633	HAND WARM
9	536366	22632	HAND WARM
10	536367	84879	ASSORTED C

At the bottom, it shows '페이지당 결과 수: 50' (Results per page: 50) and '1 - 10 (전체 10행)' (1 - 10 (Total 10 rows)).

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
Select count(*)  
From dynamic-sun-456301-m1.modulabs_project.data
```

```

4
5 Select count(*)
6 From dynamic-sun-456301-m1.modulabs_project.data
7

```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과 [결과 저장](#) [다음에서 열기](#)

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	f0_				
1	541909				

데이터 수 세기

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

```

Select
Count(InvoiceNo) COUNT_InvoiceNo,
Count(StockCode) COUNT_StockCode,
Count(Description) COUNT_Description,
Count(Quantity) COUNT_Quantity,
Count(InvoiceDate) COUNT_InvoiceDate,
Count(UnitPrice) COUNT_UnitPrice,
Count(CustomerID) COUNT_CustomerID,
Count(Country) COUNT_Country
From dynamic-sun-456301-m1.modulabs_project.data

```

8

Select

9

Count(InvoiceNo) COUNT_InvoiceNo,

10

Count(StockCode) COUNT_StockCode,

11

Count(Description) COUNT_Description,

12

Count(Quantity) COUNT_Quantity,

13

Count(InvoiceDate) COUNT_InvoiceDate,

14

Count(UnitPrice) COUNT_UnitPrice,

15

Count(CustomerID) COUNT_CustomerID,

16

Count(Country) COUNT_Country

17

From dynamic-sun-456301-m1.modulabs_project.data

18

접근성 옵션을 보려면 Alt+F1 키를

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

번호	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country
1	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(*)) AS ratio
FROM dynamic-sun-456301-m1.modulabs_project.data
UNION ALL
SELECT
    'StockCode' AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*)) AS ratio
FROM dynamic-sun-456301-m1.modulabs_project.data
UNION ALL
SELECT
    'Description' AS column_name,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*)) AS ratio
FROM dynamic-sun-456301-m1.modulabs_project.data
UNION ALL
SELECT
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*)) AS ratio
FROM dynamic-sun-456301-m1.modulabs_project.data
UNION ALL
SELECT
    'InvoiceDate' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*)) AS ratio
FROM dynamic-sun-456301-m1.modulabs_project.data
UNION ALL
SELECT
    'UnitPrice' AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*)) AS ratio
FROM dynamic-sun-456301-m1.modulabs_project.data
```

```

FROM dynamic-sun-456301-m1.modulabs_project.data
UNION ALL
SELECT
    'CustomerID' AS column_name,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / CO
FROM dynamic-sun-456301-m1.modulabs_project.data
UNION ALL
SELECT
    'Country' AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT
FROM dynamic-sun-456301-m1.modulabs_project.data

```

```

1 SELECT
2     'InvoiceNo' AS column_name,
3     ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	column_name	missing_percentage			
1	Country	0.0			
2	UnitPrice	0.0			
3	InvoiceNo	0.0			
4	Quantity	0.0			
5	InvoiceDate	0.0			
6	CustomerID	24.93			
7	Description	0.27			
8	StockCode	0.0			

결측치 처리 전략

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

```

Select distinct Description
From dynamic-sun-456301-m1.modulabs_project.data
Where StockCode = '85123A'

```

```

42
43 Select distinct Description
44 From dynamic-sun-456301-m1.modulabs_project.data
45 Where StockCode = '85123A'
46
47

```

쿼리 결과

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

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 dynamic-sun-456301-m1.modulabs_project.data
Where CustomerID is null
or Description is null;

11-4 실행 쿼리 저장 다운로드 쿼리 완료됨

```

48 Delete From dynamic-sun-456301-m1.modulabs_project.data
49 Where CustomerID is null
50 or Description is null;
51

```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과 결과 저장 다음에서 열기

작업 정보 **결과** 실행 세부정보 실행 그래프

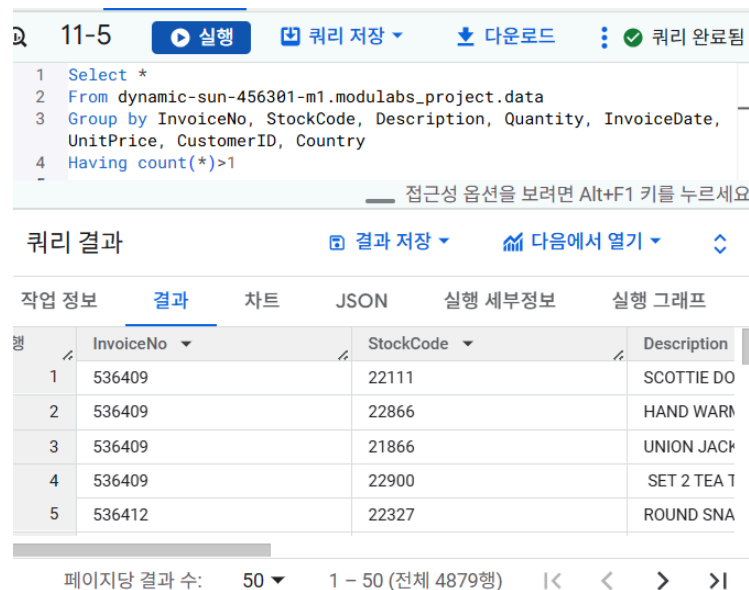
i 이 문으로 data의 행 135,080개가 삭제되었습니다. 테이블로 이동

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 *
From dynamic-sun-456301-m1.modulabs_project.data
Group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
Having count(*)>1
```



11-5 실행 쿼리 저장 다운로드 쿼리 완료됨

```
1 Select *
2 From dynamic-sun-456301-m1.modulabs_project.data
3 Group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate,
4 UnitPrice, CustomerID, Country
5 Having count(*)>1
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과 결과 저장 다음에서 열기

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceNo	StockCode	Description
1	536409	22111	SCOTTIE DO
2	536409	22866	HAND WARM
3	536409	21866	UNION JACK
4	536409	22900	SET 2 TEA T
5	536412	22327	ROUND SNA

페이지당 결과 수: 50 1 - 50 (전체 4879행) |< < > >|

중복값 처리

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

Create or Replace table dynamic-sun-456301-m1.modulabs_project.data as
Select distinct *

From dynamic-sun-456301-m1.modulabs_project.data

Group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice,



11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
With distinct_InvoiceNo as  
(Select distinct InvoiceNo  
From dynamic-sun-456301-m1.modulabs_project.data)
```

```
Select count(InvoiceNo)  
From distinct_InvoiceNo
```

1	With distinct_InvoiceNo as
2	(Select distinct InvoiceNo
3	From dynamic-sun-456301-m1.modulabs_project.data)
4	
5	Select count(InvoiceNo)
6	From distinct_InvoiceNo

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과 결과 저장 다음에서 열기

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	f0_
1	22190

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

```
Select distinct InvoiceNo
From dynamic-sun-456301-m1.modulabs_project.data
Limit 100
```

8	
9	Select distinct InvoiceNo
10	From dynamic-sun-456301-m1.modulabs_project.data
11	Limit 100
12	

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과 결과 저장 다음에서 열기

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201

페이지당 결과 수: 50 1 - 50 (전체 100행) < >

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

```
SELECT *
FROM project_name.modulabs_project.data
```



```
WHERE InvoiceNo like 'C%'
LIMIT 100;
```

```
9 Select *
10 From dynamic-sun-456301-m1.modulabs_project.data
11 Where InvoiceNo like 'C%'
12 Limit 100
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 U
2	C545329	M	Manual	-1	2011-03-01 15:47:00 U
3	C545329	M	Manual	-1	2011-03-01 15:47:00 U
4	C545330	M	Manual	-1	2011-03-01 15:49:00 U
5	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 U
6	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 U
7	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03-22 16:07:00 U
8	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12	2011-03-22 16:07:00 U
9	C547388	84050	PINK HEART SHAPE EGG FRVI	-12	2011-03-22 16:07:00 U

페이지당 결과 수: 50 1 - 50 (전체 100행) < >

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo like 'C%' THEN 1 ELSE 0 END)
FROM dynamic-sun-456301-m1.modulabs_project.data;
```

```
14 SELECT ROUND(SUM(CASE WHEN InvoiceNo like 'C%' THEN 1 ELSE 0 END)/Count(InvoiceNo)*100, 1)
15 FROM dynamic-sun-456301-m1.modulabs_project.data;
16
```

접근성

쿼리 결과

결과 저장

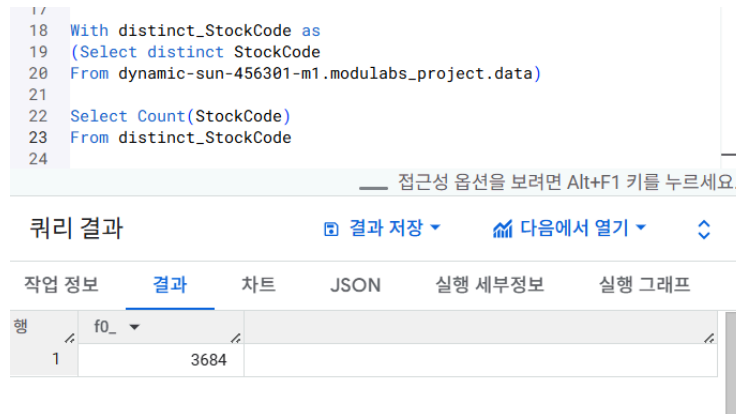
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	f0_				
1	2.2				

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
With distinct_StockCode as
(Select distinct StockCode
From dynamic-sun-456301-m1.modulabs_project.data)

Select Count(StockCode)
From distinct_StockCode
```



The screenshot shows a SQL query editor with the following code:

```
17
18 With distinct_StockCode as
19 (Select distinct StockCode
20 From dynamic-sun-456301-m1.modulabs_project.data)
21
22 Select Count(StockCode)
23 From distinct_StockCode
24
```

Below the editor, there is a toolbar with buttons: "쿼리 결과", "결과 저장", "다음에서 열기", and a refresh icon. Below the toolbar, there are tabs: "작업 정보", "결과" (selected), "차트", "JSON", "실행 세부정보", and "실행 그래프". Below the tabs, there is a table with the following data:

행	f0_
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 dynamic-sun-456301-m1.modulabs_project.data
Group by StockCode
ORDER BY sell_cnt DESC
Limit 10
```

```

25
26 SELECT StockCode, COUNT(*) AS sell_cnt
27 FROM dynamic-sun-456301-m1.modulabs_project.data
28 Group by StockCode
29 ORDER BY sell_cnt DESC
30 Limit 10
31
32

```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과 [결과 저장](#) [다음에서 열기](#)

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

	StockCode	sell_cnt
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
9	22197	1110
10	23203	1108

페이지당 결과 수: 50 1 - 10 (전체 10행) |< < > >|

- **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]', ''))
  FROM dynamic-sun-456301-m1.modulabs_project.data
)
WHERE number_count = 1
or number_count = 0

```

```

45 SELECT DISTINCT StockCode, number_count
46 FROM (
47     SELECT StockCode,
48         LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
49     FROM dynamic-sun-456301-m1.modulabs_project.data
50 )
51 WHERE number_count = 1
52 or number_count = 0
53

```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과 결과 저장 다음에서 열기

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

	StockCode	number_count
4	D	0
5	BANK CHARGES	0
6	PADS	0
7	DOT	0
8	CRUK	0

페이지당 결과 수: 50 1 - 8 (전체 8행) < >

- **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 DISTINCT StockCode, number_count
FROM (
    SELECT StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))
    FROM project_name.modulabs_project.data
)
WHERE # [[YOUR QUERY]];

```

[결과 이미지를 넣어주세요]

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM project_name.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    # [[YOUR QUERY]]
  );
```

[결과 이미지를 넣어주세요]

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM project_name.modulabs_project.data
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM project_name.modulabs_project.data
WHERE
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  # [[YOUR QUERY]] AS Description
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

UnitPrice 살펴보기

- UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```
SELECT # [[YOUR QUERY]] AS min_price, # [[YOUR QUERY]] AS max_price,
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

```
SELECT # [[YOUR QUERY]] AS cnt_quantity, # [[YOUR QUERY]] AS min_quar
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

- 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)

- **균 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(int
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID
    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 :