

# 고객을 세그멘테이션하자 [프로젝트]

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
SELECT *
FROM mainquest-426001.modulabs_project.data
LIMIT 10
```

[결과 이미지를 넣어주세요]

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
SELECT COUNT(*)
FROM mainquest-426001.modulabs_project.data
```

[결과 이미지를 넣어주세요]

### 데이터 수 세기

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

```
# [[YOUR QUERY]]
SELECT
  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 mainquest-426001.modulabs_project.data
```

[결과 이미지를 넣어주세요]

## 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

```
# [[YOUR QUERY]]
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM mainquest-426001.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_percent
FROM mainquest-426001.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_perce
FROM mainquest-426001.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_percenta
FROM mainquest-426001.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_perce
FROM mainquest-426001.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_percent
FROM mainquest-426001.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_perce
FROM mainquest-426001.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_percentag
FROM mainquest-426001.modulabs_project.data

```

[결과 이미지를 넣어주세요]

## 결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM mainquest-426001.modulabs_project.data
WHERE StockCode = '85123A'

```

[결과 이미지를 넣어주세요]

## 결측치 처리

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

```

DELETE FROM mainquest-426001.modulabs_project.data
WHERE Description IS NULL
;
DELETE FROM mainquest-426001.modulabs_project.data
WHERE CustomerID 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
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*)
FROM mainquest-426001.modulabs_project.data
GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
HAVING COUNT(*) > 1;
```

[결과 이미지를 넣어주세요]

## 중복값 처리

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

```
# [[YOUR QUERY]]
CREATE OR REPLACE TABLE mainquest-426001.modulabs_project.data2 AS
SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
FROM mainquest-426001.modulabs_project.data
GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country;
```

[결과 이미지를 넣어주세요]

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT InvoiceNo)
FROM mainquest-426001.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
# [[YOUR QUERY]]
SELECT DISTINCT InvoiceNo
FROM mainquest-426001.modulabs_project.data2
LIMIT 100
```

[결과 이미지를 넣어주세요]

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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]]
LIMIT 100;
```

[결과 이미지를 넣어주세요]

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

```
SELECT ROUND(SUM(CASE WHEN # [[YOUR QUERY]] THEN 1 ELSE 0 END)/ # [[YOUR QUERY]], 1)
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

### StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT StockCode)
FROM mainquest-426001.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

쿼리 결과  결과 저장 

	작업 정보	결과	차트	JSON
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 mainquest-426001.modulabs_project.data2
GROUP BY 1
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

<	작업 정보	결과	차트	JSON	실행
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 mainquest-426001.modulabs_project.data2
)
WHERE number_count = 1 OR number_count = 0
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

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

- 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(COUNT(CASE WHEN StockCode = 'POST' THEN 1
  WHEN StockCode = 'M' THEN 1
  WHEN StockCode = 'PADS' THEN 1
  WHEN StockCode = 'D' THEN 1
  WHEN StockCode = 'BANK CHARGES' THEN 1
  WHEN StockCode = 'DOT' THEN 1
  WHEN StockCode = 'CRUK' THEN 1
  WHEN StockCode = 'C2' THEN 1
  ELSE NULL END) * 100
  /COUNT(StockCode), 2) AS percentage
FROM mainquest-426001.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

쿼리 결과		결과 저장	데이터 탐색
<	작업 정보	결과	차트
JSON	실행 세부정보		
행	percentage		
1	0.48		

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM mainquest-426001.modulabs_project.data2
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode, CASE
      WHEN StockCode IN ('POST', 'M', 'PADS', 'D', 'BANK CHARGES', 'DOT', 'CRUK', 'C2') THEN StockCo
      ELSE NULL
    END AS destock
    FROM mainquest-426001.modulabs_project.data2
  )
  WHERE destock IS NOT NULL
);
```

[결과 이미지를 넣어주세요]

쿼리 결과		결과 저장	데이터 탐색
작업 정보	결과	실행 세부정보	실행 그래프
<div> <span>이 문으로 data2의 행 1,915개가 삭제되었습니다.</span> <span>테이블로 이동</span> </div>			

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM mainquest-426001.modulabs_project.data2
GROUP BY 1
LIMIT 30
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 데이터 탐색

작업 정보			결과	차트	JSON	실행 세부정보
행	Description	description_cnt				
1	SET OF 4 KNICK KNACK TINS ...	328				
2	FELTCRAFT PRINCESS OLIVIA ...	254				
3	FELTCRAFT PRINCESS LOLA D...	383				
4	EMBROIDERED RIBBON REEL E...	87				
5	EMBROIDERED RIBBON REEL S...	63				
6	PINK BLUE FELT CRAFT TRINK...	465				
7	JAM MAKING SET WITH JARS	966				
8	ASSORTED COLOUR MINI CAS...	372				
9	SCANDINAVIAN REDS RIBBONS	343				
10	PAPER CHAIN KIT VINTAGE C...	718				
11	TRADITIONAL KNITTING NANCY	523				

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM mainquest-426001.modulabs_project.data2
WHERE description = 'High Resolution Image' OR description = 'Next Day Carriage'
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 데이터 탐색

작업 정보 **결과** 실행 세부정보

**i** 이 문으로 data2의 행 83개가 삭제되었습니다.

[테이블로 이동](#)

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE mainquest-426001.modulabs_project.data2 AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM mainquest-426001.modulabs_project.data2;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 데이터 탐색

작업 정보 **결과** 실행 세부정보 실행 그:

**i** 이 문으로 이름이 data2인 테이블이 교체되었습니다.

[테이블로 이동](#)

## UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT
    MIN(UnitPrice) AS min_price
    , MAX(UnitPrice) AS max_price
    , AVG(UnitPrice) AS avg_price
FROM mainquest-426001.modulabs_project.data2;
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	차트	JSON	
행	min_price	max_price	avg_price		
1	0.0	649.5	2.904956757405...		

- 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
    , AVG(Quantity) AS avg_quantity
FROM mainquest-426001.modulabs_project.data2
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프
행	cnt_quantity ▾	min_quantity ▾	max_quantity ▾	avg_quantity ▾		
1	33	1	12540	420.5151515151...		

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE mainquest-426001.modulabs_project.noerrdata AS
SELECT *
FROM mainquest-426001.modulabs_project.data2
WHERE UnitPrice != 0
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	실행 세부정보
	작업 정보	결과	실행 세부정보	
<p><b>i</b> 이 문으로 이름이 noerrdata인 테이블이 교체되었습니다</p> <p>테이블로 이동</p>				

## 11-7. RFM 스코어

### Recency





- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT FORMAT_DATE('%Y-%m-%d', InvoiceDate) AS InvoiceDay, *
FROM mainquest-426001.modulabs_project.noerrdata;

--자료 replace함
CREATE OR REPLACE TABLE mainquest-426001.modulabs_project.noerrdata AS
SELECT
    InvoiceNo
    , StockCode
    , Quantity
    , FORMAT_DATE('%Y-%m-%d', InvoiceDate) AS InvoiceDate
    , UnitPrice
    , CustomerID
    , Country
    , Description
FROM mainquest-426001.modulabs_project.noerrdata;
```

[결과 이미지를 넣어주세요]

쿼리 결과  결과 저장 



< 결과 차트 JSON 실행 세부정보

행	InvoiceDay	InvoiceNo
1	2011-11-03	574301
2	2011-11-03	574301
3	2011-11-03	574301
4	2011-11-03	574301
5	2011-11-03	574301
6	2011-11-03	574301
7	2011-11-03	574301
8	2011-11-03	574301
9	2011-11-03	574301

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
    (SELECT MAX(InvoiceDate) FROM mainquest-426001.modulabs_project.noerrdata) AS most_recent_date,
    InvoiceDate AS InvoiceDay,
    *
FROM mainquest-426001.modulabs_project.noerrdata;
```

[결과 이미지를 넣어주세요]

< 쿼리 결과  

< 작업 정보 결과 차트 JSON

행	most_recent_date	InvoiceDay
1	2011-12-09	2010-12-01
2	2011-12-09	2010-12-01
3	2011-12-09	2010-12-03
4	2011-12-09	2010-12-05
5	2011-12-09	2010-12-06
6	2011-12-09	2010-12-08
7	2011-12-09	2010-12-09
8	2011-12-09	2010-12-09
9	2011-12-09	2010-12-09

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
    CustomerID,
    MAX(InvoiceDate) AS InvoiceDay
FROM mainquest-426001.modulabs_project.noerrdata
GROUP BY 1
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보 결과 차트 JSON

행	CustomerID	InvoiceDay
1	12583	2011-12-07
2	16098	2011-09-13
3	17967	2010-12-03
4	13069	2011-12-09
5	17677	2011-12-08
6	15529	2011-11-17
7	16795	2010-12-09
8	12872	2011-01-17
9	12682	2011-12-06
10	14713	2011-11-30

페이지당 결과 수: 50

- 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(InvoiceDate) AS InvoiceDay
    FROM project_name.modulabs_project.data
    GROUP BY CustomerID
)
ORDER BY recency
;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보 결과 차트 데이터

행	CustomerID	recency
1	13113	0
2	13069	0
3	12526	0
4	12713	0
5	15804	0
6	12985	0
7	17428	0
8	16558	0

- 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 **user\_r**이라는 이름의 테이블로 저장하기

```
CREATE OR REPLACE TABLE mainquest-426001.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 mainquest-426001.modulabs_project.noerrdata
  GROUP BY CustomerID
)
ORDER BY recency
```

[결과 이미지를 넣어주세요]

### 쿼리 결과

작업 정보   **결과**   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 user\_r인 새 테이블이 생성되었습니다.

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID
  , COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM mainquest-426001.modulabs_project.noerrdata
GROUP BY 1
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터

<

작업 정보

결과

차트

JSON

실행

행	CustomerID	purchase_cnt
1	12583	17
2	16098	7
3	17967	1
4	13069	27
5	17677	43
6	15529	12
7	16795	1
8	12872	2

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM mainquest-426001.modulabs_project.noerrdata
Group by 1
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 데이터

<	작업 정보	결과	차트	JSON	실행
행	CustomerID	item_cnt			
1	12583	4978			
2	16098	613			
3	17967	73			
4	13069	5454			
5	17677	9722			
6	15529	3447			
7	16795	239			
8	12872	319			
9	17682	5485			

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

```
CREATE OR REPLACE TABLE mainquest-426001.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID
    , COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM mainquest-426001.modulabs_project.noerrdata
  GROUP BY 1
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM mainquest-426001.modulabs_project.noerrdata
  Group by 1
)

-- 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 mainquest-426001.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

user_rf				
스키마 세부정보 미리보기 계보 데이터 프로				
행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	18010	1	60	256
3	12792	1	215	256
4	15083	1	38	256
5	15520	1	314	1
6	13436	1	76	1
7	13298	1	96	1
8	14569	1	79	1
9	13357	1	321	257
10	14476	1	110	257
11	15471	1	256	2
12	15195	1	1404	2
13	14204	1	72	2
14	12650	1	250	3
15	14578	1	240	3
16	15318	1	642	3
17	16528	1	171	3
18	16569	1	93	3
19	12478	1	233	3
20	17914	1	457	3
21	12442	1	181	3
22	15992	1	17	3
23	14536	1	39	259
24	17383	1	148	4

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice), 0) AS user_total
FROM mainquest-426001.modulabs_project.noerrdata
Group by 1
;
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	데이터
작업 정보			결과	차트
JSON	실			
행	CustomerID	user_total		
1	12583	503.0		
2	16098	296.0		
3	17967	81.0		
4	13069	809.0		
5	17677	886.0		
6	15529	1148.0		
7	16795	146.0		

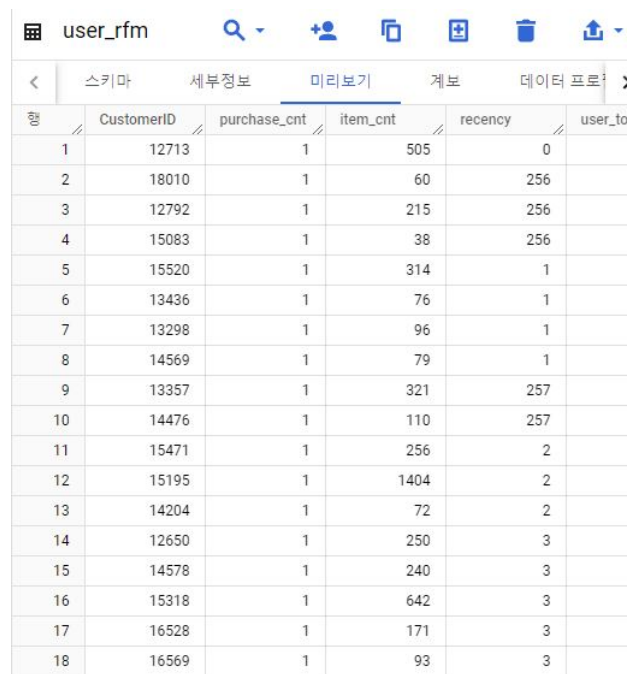
- 고객별 평균 거래 금액 계산
  - 고객별 평균 거래 금액을 구하기 위해 1) `data` 테이블을 `user_rf` 테이블과 조인(LEFT JOIN) 한 후, 2) `purchase_cnt` 로 나누어서 3) `user_rf_m` 테이블로 저장하기

```

CREATE OR REPLACE TABLE mainquest-426001.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(ut.user_total/rf.purchase_cnt, 0) AS user_average
FROM mainquest-426001.modulabs_project.user_rf AS rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice), 0) AS user_total
  FROM mainquest-426001.modulabs_project.noerrdata
  Group by 1
) AS ut
ON rf.CustomerID = ut.CustomerID
ORDER BY 2

```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_to
1	12713	1	505	0	
2	18010	1	60	256	
3	12792	1	215	256	
4	15083	1	38	256	
5	15520	1	314	1	
6	13436	1	76	1	
7	13298	1	96	1	
8	14569	1	79	1	
9	13357	1	321	257	
10	14476	1	110	257	
11	15471	1	256	2	
12	15195	1	1404	2	
13	14204	1	72	2	
14	12650	1	250	3	
15	14578	1	240	3	
16	15318	1	642	3	
17	16528	1	171	3	
18	16569	1	93	3	

## RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```

# [[YOUR QUERY]]
SELECT *
FROM mainquest-426001.modulabs_project.user_rfm

```

[결과 이미지를 넣어주세요]

user_rfm							
<span>쿼리</span> <span>공유</span> <span>복사</span> <span>스냅샷</span> <span>삭제</span> <span>내보내기</span>							
<span>스키마</span> <span>세부정보</span> <span>미리보기</span> <span>계보</span> <span>데이터 프로필</span> <span>데이터 품질</span>							
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	
1	12713	1	505	0	77.0	77.0	
2	18010	1	60	256	70.0	70.0	
3	12792	1	215	256	60.0	60.0	
4	15083	1	38	256	33.0	33.0	
5	15520	1	314	1	31.0	31.0	
6	13436	1	76	1	70.0	70.0	
7	13298	1	96	1	8.0	8.0	
8	14569	1	79	1	47.0	47.0	
9	13357	1	321	257	262.0	262.0	
10	14476	1	110	257	60.0	60.0	

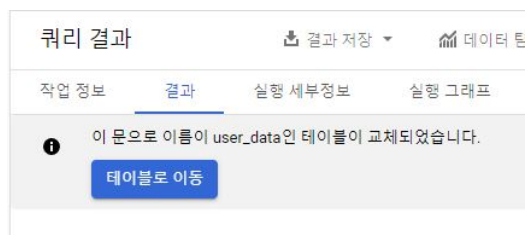
## 11-8. 추가 Feature 추출

### 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(interval_), 2) END AS average_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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]

user\_data

🔍 쿼리

👤 공유

📄 복사

📄 스냅샷

🗑️ 삭제

📤 내보내기

🔄 새로고침

스키마

세부정보

미리보기

계보

데이터 프로필

데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	15118	1	1440	134	0.0	0.0	1	0.0
2	13366	1	144	50	0.0	0.0	1	0.0
3	17752	1	192	359	0.0	0.0	1	0.0
4	16953	1	10	30	2.0	2.0	1	0.0
5	18174	1	50	7	2.0	2.0	1	0.0
6	12814	1	48	101	2.0	2.0	2	0.0

### 3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기
  - 취소 빈도(cancel\_frequency) : 고객 별로 취소한 거래의 총 횟수
  - 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 mainquest-426001.modulabs_project.final_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    SUM(if(InvoiceNO LIKE 'C%', 1, 0 )) AS cancel_frequency
  FROM mainquest-426001.modulabs_project.noerrdata
  GROUP BY 1
)

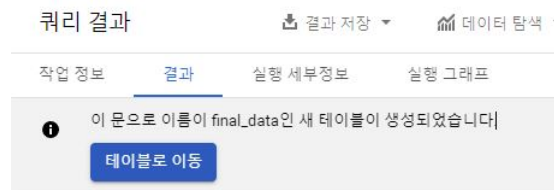
SELECT u.*
, t.* EXCEPT(CustomerID)
, t.cancel_frequency / t.total_transactions AS cancel_rate
FROM `mainquest-426001.modulabs_project.user_data` AS u

```



```
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
```

[결과 이미지를 넣어주세요]



- 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 **user\_data**를 출력하기

```
# [[YOUR QUERY]];
SELECT *
FROM mainquest-426001.modulabs_project.final_data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 | 데이터 탐색

작업 정보 | **결과** | 실행 세부정보 | 실행 그래프

	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_r
1 26	2.0	2.0	1	0.0	1	0	
2 36	4.0	4.0	1	0.0	1	0	
3 15	1.0	1.0	1	0.0	1	0	
4 30	2.0	2.0	1	0.0	1	0	
5 2	3.0	3.0	1	0.0	1	0	
6 17	7.0	7.0	1	0.0	1	0	
7 42	1.0	1.0	1	0.0	1	0	