# 고객을 세그먼테이션하자 [프로젝트]

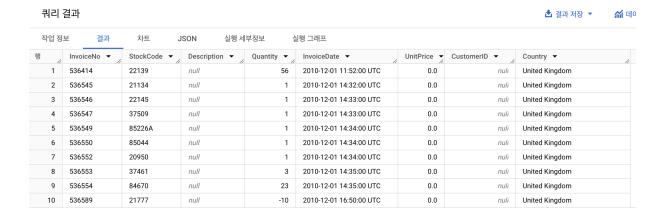
## 11-2. 데이터 불러오기

### 데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*) AS row_count
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



### 데이터 수 세기

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

```
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 lunar-planet-425501-t3.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

```
SELECT
  'InvoiceNo' AS InvoiceNo,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
 'StockCode' AS StockCode,
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'Description' AS Description,
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'Quantity' AS Quantity,
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'InvoiceDate' AS InvoiceDate,
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
```

```
SELECT

'UnitPrice' AS UnitPrice,

ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage

FROM lunar-planet-425501-t3.modulabs_project.data

UNION ALL

SELECT

'CustomerID' AS CustomerID,

ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage

FROM lunar-planet-425501-t3.modulabs_project.data

UNION ALL

SELECT

'Country' AS Country,

ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage

FROM lunar-planet-425501-t3.modulabs_project.data;
```

### 쿼리 결과

작업 정	보 결과	차트	JSON {	실행 세부정보
행 //	InvoiceNo ▼	/	missing_perce	entage
1	CustomerID		2	4.93
2	StockCode			0.0
3	Country			0.0
4	Description			0.27
5	UnitPrice			0.0
6	InvoiceNo			0.0
7	Quantity			0.0
8	InvoiceDate			0.0

#### 결측치 처리 전략

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

```
SELECT DISTINCT Description

FROM lunar-planet-425501-t3.modulabs_project.data

WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

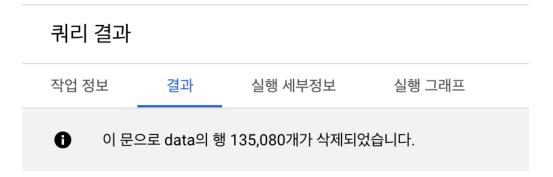
즈	<b>ነ</b> 업 정	보	결과	차트	,	JSON	실행 세부정5
행	11	Desci	ription 🔻		//		
	1	?					
	2	wrong	gly marked c	arton 22804			
	3	CREA	M HANGING	HEART T-LIC	<b>3</b>		
	4	WHIT	E HANGING	HEART T-LIG			

### 결측치 처리

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

DELETE FROM lunar-planet-425501-t3.modulabs\_project.data
WHERE Description IS NULL
OR 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

```
WITH duplicates AS (
    SELECT (CASE WHEN COUNT(*) > 1 THEN 1 ELSE 0 END) as duplicate
    FROM lunar-planet-425501-t3.modulabs_project.data
    GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
)
SELECT SUM(duplicate) as dup_count
FROM duplicates;
```



### 중복값 처리

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

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.modulabs_project.data AS
SELECT DISTINCT *
FROM lunar-planet-425501-t3.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\_InvoiceNo FROM lunar-planet-425501-t3.modulabs\_project.data;

[결과 이미지를 넣어주세요]





• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo
FROM lunar-planet-425501-t3.modulabs\_project.data
LIMIT 100;

[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.data
WHERE InvoiceNo LIKE "C%"
LIMIT 100;
```

[결과 이미지를 넣어주세요]



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

SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE "C%" THEN 1 ELSE 0 END)/ COUNT(InvoiceNo) \* 100, 1) AS Pct\_Car FROM lunar-planet-425501-t3.modulabs\_project.data;

[결과 이미지를 넣어주세요]





### StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode) AS Unique\_StockCode FROM lunar-planet-425501-t3.modulabs\_project.data;

[결과 이미지를 넣어주세요]

# 쿼리 결과



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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

# 쿼리 결과

작업 정	보 결과	차트	JSON	실행 세부정5
행 //	StockCode -	,	sell_cnt	<b>▼</b>
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 lunar-planet-425501-t3.modulabs_project.data
)
WHERE number_count = 1
    OR number_count = 0;
```

작업 정	보 결과	차트	JSON	실행 세부장
행 //	StockCode ▼	//	number_cour	nt ▼
1	POST			0
2	М			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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
WITH StockCodes AS (

SELECT StockCode, number_count

FROM (

SELECT StockCode,

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

FROM lunar-planet-425501-t3.modulabs_project.data
)
)

SELECT ROUND(SUM(CASE WHEN number_count < 5 THEN 1 ELSE 0 END)/ COUNT(StockCode) * 100, 2) AS StockCode_
FROM StockCodes;

"""

OR

SELECT ROUND(SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) < 5 THEN FROM lunar-planet-425501-t3.modulabs_project.data;
"""
```

[결과 이미지를 넣어주세요]

작업 정	성보	결과	차트
행	Stocl	kCode_Outli	erRatio 🔻
1			0.48

#### • 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM lunar-planet-425501-t3.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 lunar-planet-425501-t3.modulabs_project.data
    )
    WHERE number_count < 5
);

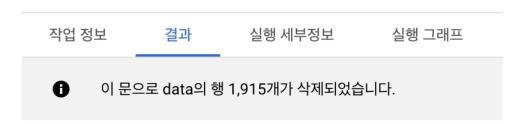
UITOR

DELETE FROM lunar-planet-425501-t3.modulabs_project.data
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) < 5

UITOR
```

[결과 이미지를 넣어주세요]

# 쿼리 결과



### Description 살펴보기

#### • 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY Description
LIMIT 30;
```

#### [결과 이미지를 넣어주세요]



#### • 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM lunar-planet-425501-t3.modulabs_project.data
WHERE Description = 'Next Day Carriage'
OR Description = 'High Resolution Image';
```

[결과 이미지를 넣어주세요]



#### • 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.modulabs_project.data AS

SELECT

* EXCEPT (Description),

UPPER(Description) AS Description

FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보 결과 실행 세부정보 실행 그래프

0

이 문으로 이름이 data인 테이블이 교체되었습니다.

### UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```
SELECT
MIN(UnitPrice) AS min_price,
MAX(UnitPrice) AS max_price,
AVG(UnitPrice) AS avg_price
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

### 쿼리 결과

작업 정	보	결과		차트	JSON	실행 세부정보
행 //	min_	price 🔻	11	max_price	<b>▼</b>	avg_price ▼
1			0.0		649.5	2.904956757406

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

```
SELECT
COUNT(Quantity) AS cnt_quantity,
MIN(Quantity) AS min_quantity,
MAX(Quantity) AS max_quantity,
AVG(Quantity) AS avg_quantity
FROM lunar-planet-425501-t3.modulabs_project.data
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

## ← 쿼리 결과

작업 정	보	결과	차트	JSON	실행 세부정보	실행 그래프
행 //	cnt_	quantity -	min_q	uantity 🔻	max_quantity ▼	avg_quantity ▼
1		33		1	12540	420.5151515151

• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE lunar-planet-425501-t3.modulabs\_project.data AS

SELECT \*

FROM lunar-planet-425501-t3.modulabs\_project.data

WHERE UnitPrice != 0;

[결과 이미지를 넣어주세요]

# 쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프 이 문으로 이름이 data인 테이블이 교체되었습니다.

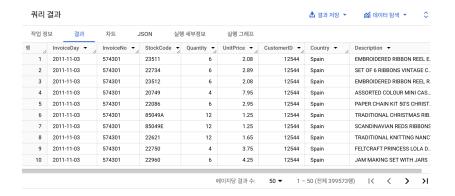
## 11-7. RFM 스코어

### Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay, \*
FROM lunar-planet-425501-t3.modulabs\_project.data;

[결과 이미지를 넣어주세요]



#### • 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT

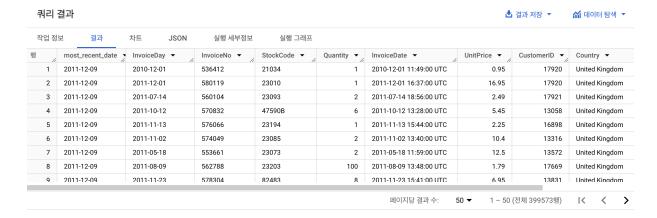
MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,

DATE(InvoiceDate) AS InvoiceDay,

*

FROM lunar-planet-425501-t3.modulabs_project.data;
```

#### [결과 이미지를 넣어주세요]



### • 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY CustomerID;
```

[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 <u>user\_r</u> 이라는 이름의 테이블로 저장하기

```
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY CustomerID
)
);
[결과 이미지를 넣어주세요]
```

작업 정보 결과 실행 세부정보 실행 그래프

0

이 문으로 이름이 user\_r인 새 테이블이 생성되었습니다.

### **Frequency**

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY 1;
```

[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
COUNT(StockCode) AS item_cnt
```

```
GROUP BY 1;
[결과 이미지를 넣어주세요]
   쿼리 결과
                                                                                  🏜 결과 저장 ▼
   작업 정보
               결과
                        차트
                                 JSON
                                           실행 세부정보
                                                          실행 그래프
         CustomerID -
                        item_cnt ▼
     1
                 12544
                                   19
     2
                 13568
                                   43
     3
                 13824
                                   46
     4
                 14080
                                    4
```

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

#### • 전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 user\_rf 라는 이름의 테이블에 저장하기

90

158

69

13

108

89

33

FROM lunar-planet-425501-t3.modulabs\_project.data

14336

14592

15104

15360

15872

16128

1638/

5

6

7

8

9

10

11

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.modulabs_project.user_rf AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
 SELECT
   CustomerID,
   COUNT(DISTINCT InvoiceNo) AS purchase_cnt
 FROM lunar-planet-425501-t3.modulabs_project.data
 GROUP BY 1
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
 SELECT
   CustomerID,
   COUNT(StockCode) AS item_cnt
 FROM lunar-planet-425501-t3.modulabs_project.data
 GROUP BY 1
-- 기존의 user_r에 (1)과 (2)를 통합
 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 lunar-planet-425501-t3.modulabs_project.user_r AS ur

ON pc.CustomerID = ur.CustomerID;
```

# 쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

0

이 문으로 이름이 user\_rf인 새 테이블이 생성되었습니다.

### Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
   CustomerID,
   ROUND(SUM(Quantity * UnitPrice),1) AS user_total
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과 저장 ▼

작업 정보	<b>크</b> 결과	차트	JSON	실행 세부정보	실행 그래프
행 //	CustomerID ▼	user_total	<b>▼</b>		
1	12544		299.7		
2	13568		187.1		
3	13824		1698.9		
4	14080		45.6		
5	14336		1614.9		
6	14592		557.9		
7	15104		968.6		
8	15360		427.9		
9	15872		316.2		

고객을 세그먼테이션하자 [프로젝트] 18

페이지당 결과 수:

50 ▼

1 - 50 (전체 4362행)

- 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 lunar-planet-425501-t3.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) AS user_average
FROM lunar-planet-425501-t3.modulabs_project.user_rf rf
LEFT JOIN (
 -- 고객 별 총 지출액
 SELECT
   CustomerID,
   ROUND(SUM(Quantity * UnitPrice)) AS user_total
 FROM lunar-planet-425501-t3.modulabs_project.data
 GROUP BY 1
) ut
ON rf.CustomerID = ut.CustomerID;
```

# 쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

0

이 문으로 이름이 user\_rfm인 새 테이블이 생성되었습니다.

### RFM 통합 테이블 출력하기

• 최종 user\_rfm 테이블을 출력하기

```
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

1	CustomerID ▼	purchase_cnt ▼	item_cnt ▼	recency ▼	user_total ▼	user_average ▼
1	14090	1	1	324	76.0	76.0
2	16148	1	1	296	76.0	76.0
3	18141	1	1	360	-35.0	-35.0
4	12791	1	1	373	178.0	178.0
5	12943	1	1	301	-4.0	-4.0
6	17291	1	1	308	551.0	551.0
7	15562	1	1	351	135.0	135.0
8	13703	1	1	318	100.0	100.0

## 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 lunar-planet-425501-t3.modulabs_project.user_data AS
WITH unique_products AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT StockCode) AS unique_products
        FROM lunar-planet-425501-t3.modulabs_project.data
        GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM lunar-planet-425501-t3.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

## 쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프 이 문으로 이름이 user\_data인 새 테이블이 생성되었습니다.

### 2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
  - 평균 구매 소요 일수를 계산하고, 그 결과를 user\_data 에 통합

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.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
   -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS
     lunar-planet-425501-t3.modulabs_project.data
   WHERE CustomerID IS NOT NULL
 GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM lunar-planet-425501-t3.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

## 쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

## 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 lunar-planet-425501-t3.modulabs_project.user_data AS
WITH TransactionInfo AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT InvoiceNo) AS total_transactions,
        SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
        FROM lunar-planet-425501-t3.modulabs_project.data
        GROUP BY 1
)

SELECT
        u.*,
        t.* EXCEPT(CustomerID),
        ROUND((t.cancel_frequency / t.total_transactions * 100), 2) AS cancel_rate
FROM `lunar-planet-425501-t3.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

## 쿼리 결과

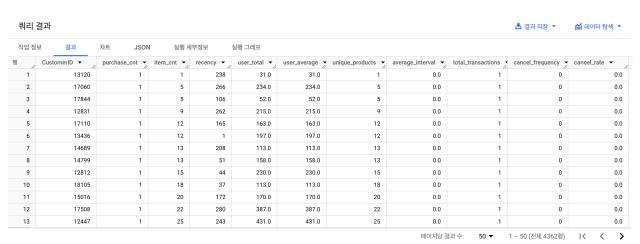
작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

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

```
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]



#### [회고]

- LMS에서 설명된 내용을 따라하면서 지금까지 배웠던 리눅스, 깃헙, SQL을 사용하면서 복습할 수 있었습니다.
- 지난 노드에서 배운 RFM의 주 축인 recency, frequency, monetary외에도 구매하는 제품군의 다양성, 평균 구매 주기, 구매 취소 경향성을 통해 고객군을 더 세분화해서 나눌 수 있음을 배웠습니다.
- 앞에서 좀 더 간단하게 SQL을 사용할 수 있었는데, 노드 상에서 # [[YOUR QUERY]]가 있는 부분을 수정하는 것이다보니 몇 개는 복잡한 버전과 좀 더 간단한 버전 두가지로 적어봤습니다.
- 내일 새로 배울 노드에서 어떻게 파이썬을 사용해 데이터를 가시화할 수 있을지 기대됩니다.