# 고객을 세그먼테이션하자 [프로젝트] (1)

# 11-2. 데이터 불러오기

# 데이터 살펴보기

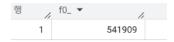
• 테이블에 있는 10개의 행만 출력하기

SELECT \*
FROM velvety-study-473605-t0.modulabs\_project.data
LIMIT 10



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT COUNT(\*)
FROM velvety-study-473605-t0.modulabs\_project.data



#### 데이터 수 세기

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

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 velvety-study-473605-t0.modulabs\_project.data



# 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) * 100 / COUNT(*), 2) AS missing_percentage
   velvety-study-473605-t0.modulabs_project.data
 UNION ALL
 SELECT
   'StockCode' AS column_name,
   ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
  velvety-study-473605-t0.modulabs_project.data
 UNION ALL
 SELECT
   'Description' AS column_name,
   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
   velvety-study-473605-t0.modulabs_project.data
 UNION ALL
 SELECT
   'Quantity' AS column_name,
   ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
   velvety-study-473605-t0.modulabs_project.data
 UNION ALL
 SELECT
   'InvoiceDate' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
   velvety-study-473605-t0.modulabs_project.data
 UNION ALL
 SELECT
   'UnitPrice' AS column_name,
   ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
 FROM
   velvety-study-473605-t0.modulabs_project.data
 UNION ALL
 SELECT
   'CustomerID' AS column_name,
   ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
 FROM
   velvety-study-473605-t0.modulabs_project.data
 UNION ALL
```

**SELECT** 

'Country' AS column\_name,

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

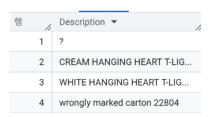
velvety-study-473605-t0.modulabs\_project.data

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

# 결측치 처리 전략

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

SELECT DISTINCT Description
FROM velvety-study-473605-t0.modulabs\_project.data
WHERE StockCode = '85123A'



# 결측치 처리

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

DELETE FROM velvety-study-473605-t0.modulabs\_project.data WHERE CustomerID IS NULL OR Description IS NULL;



# 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 num_duplicate_groups
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  FROM
    velvety-study-473605-t0.modulabs_project.data
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
```

Quantity,
InvoiceDate,
UnitPrice,
CustomerID,
Country
HAVING
COUNT(\*) > 1



# 중복값 처리

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

CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs\_project.data\_cleaned AS
SELECT DISTINCT \*
FROM velvety-study-473605-t0.modulabs\_project.data
WHERE
CustomerID IS NOT NULL
AND Description IS NOT NULL;

(실행두번눌려서 data\_cleaned로나오네요..)

이 문으로 이름이 data\_cleaned인 테이블이 교체되었습니다.

# 11-6. 데이터 전처리(3): 오류값 처리

# InvoiceNo 살펴보기

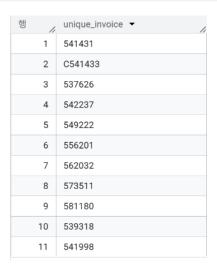
• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT COUNT(DISTINCT InvoiceNo) AS unique\_invoice\_count FROM velvety-study-473605-t0.modulabs\_project.data\_cleaned;



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo AS unique\_invoice FROM velvety-study-473605-t0.modulabs\_project.data\_cleaned LIMIT 100;



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

SELECT \*

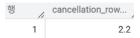
FROM velvety-study-473605-t0.modulabs\_project.data\_cleaned WHERE InvoiceNo LIKE 'C%' LIMIT 100;



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

SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*),1) AS cancellation\_row\_rat e\_percentage

FROM velvety-study-473605-t0.modulabs\_project.data\_cleaned;



#### StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode) AS unique\_stockCode FROM velvety-study-473605-t0.modulabs\_project.data\_cleane



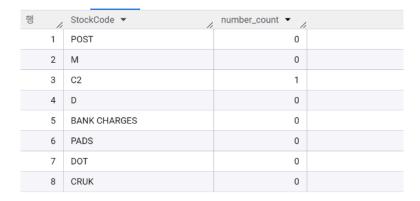
- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기
  - 。 상위 10개의 제품들을 출력하기

SELECT StockCode, COUNT(\*) AS sell\_cnt FROM velvety-study-473605-t0.modulabs\_project.data\_cleaned GROUP BY StockCode ORDER BY sell\_cnt DESC LIMIT 10;

| 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 velvety-study-473605-t0.modulabs_project.data_cleaned
)
WHERE number_count IN (0,1)
```



• 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)

```
SELECT
DISTINCT StockCode,
number_count
FROM
(SELECT
StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM
velvety-study-473605-t0.modulabs_project.data_cleaned
)
WHERE number_count IN (0, 1);
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM velvety-study-473605-t0.modulabs_project.data_cleaned

WHERE StockCode IN (

SELECT DISTINCT StockCode

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

FROM velvety-study-473605-t0.modulabs_project.data_cleaned
)

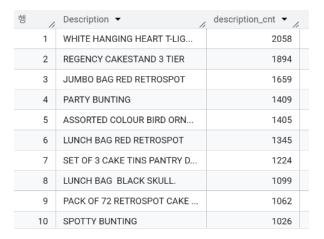
WHERE number_count IN (0, 1)
);
```

작업 정보 결과 실행 세부정보 실행 그래프 \_\_\_\_\_\_ 이 문으로 data\_cleaned의 행 1,915개가 삭제되었습니다.

# Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT Description,COUNT(\*) AS description\_cnt FROM velvety-study-473605-t0.modulabs\_project.data\_cleaned GROUP BY Description ORDER BY description\_cnt DESC LIMIT 30;



• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE
FROM velvety-study-473605-t0.modulabs\_project.data\_cleaned
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 data\_cleaned의 행 83개가 삭제되었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs\_project.data\_standardized AS
SELECT

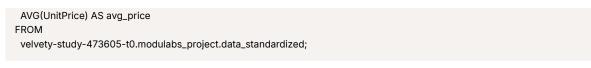
\* EXCEPT (Description),
UPPER(Description) AS Description
FROM
velvety-study-473605-t0.modulabs\_project.data\_cleaned;

● 이 문으로 이름이 data\_standardized인 새 테이블이 생성되었습니다.

# UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

SELECT
MIN(UnitPrice) AS min\_price,
MAX(UnitPrice) AS max\_price,





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

SESELECT
COUNT(\*) AS cnt\_quantity,
MIN(Quantity) AS min\_quantity,
MAX(Quantity) AS max\_quantity,
AVG(Quantity) AS avg\_quantity
FROM
velvety-study-473605-t0.modulabs\_project.data\_standardized
WHERE
UnitPrice = 0;



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs\_project.data AS SELECT \*
FROM velvety-study-473605-t0.modulabs\_project.data\_standardized
WHERE UnitPrice > 0;

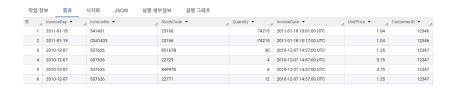
● 이 문으로 이름이 data인 테이블이 교체되었습니다.

# 11-7. RFM 스코어

#### Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay,\* FROM velvety-study-473605-t0.modulabs\_project.data;



#### • 가장 최근 구매 일자를 MAX() 함수로 찾아보기

**SELECT** 

MAX(DATE(InvoiceDate)) OVER () AS most\_recent\_date, DATE(InvoiceDate) AS InvoiceDay,\*
FROM velvety-study-473605-t0.modulabs\_project.data;



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

**SELECT** 

CustomerID,

MAX(DATE(InvoiceDate)) AS InvoiceDay FROM velvety-study-473605-t0.modulabs\_project.data GROUP BY CustomerID;

| 행 // | CustomerID 🔻 | ,    | InvoiceDay ▼ |
|------|--------------|------|--------------|
| 1    | 1            | 2346 | 2011-01-18   |
| 2    | 1            | 2347 | 2011-12-07   |
| 3    | 1            | 2348 | 2011-09-25   |
| 4    | 1            | 2349 | 2011-11-21   |
| 5    | 1            | 2350 | 2011-02-02   |
| 6    | 1            | 2352 | 2011-11-03   |
| 7    | 1            | 2353 | 2011-05-19   |
| 8    | 1            | 2354 | 2011-04-21   |
| 9    | 1            | 2355 | 2011-05-09   |
| 10   | 1            | 2356 | 2011-11-17   |
| 11   | 1            | 2357 | 2011-11-06   |

• 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 velvety-study-473605-t0.modulabs_project.data
GROUP BY CustomerID
);
```

| ±υ   | 0            |           |
|------|--------------|-----------|
| 행 // | CustomerID ▼ | recency ▼ |
| 1    | 12539        | 22        |
| 2    | 12579        | 73        |
| 3    | 12604        | 79        |
| 4    | 12621        | 1         |
| 5    | 12625        | 211       |
| 6    | 12865        | 26        |
| 7    | 12977        | 156       |

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

```
CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs_project.user_r AS
SELECT
CustomerID,
DATE_DIFF(MAX(InvoiceDay) OVER (), InvoiceDay, DAY) AS Recency
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM
velvety-study-473605-t0.modulabs_project.data
GROUP BY
CustomerID
);
```

**⑥** 이 문으로 이름이 user\_r인 새 테이블이 생성되었습니다.

# **Frequency**

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM
velvety-study-473605-t0.modulabs_project.data
GROUP BY
CustomerID;
```

| 행 // | CustomerID ▼ | purchase_cnt ▼ |
|------|--------------|----------------|
| 1    | 12346        | 2              |
| 2    | 12347        | 7              |
| 3    | 12348        | 4              |
| 4    | 12349        | 1              |
| 5    | 12350        | 1              |
| 6    | 12352        | 8              |
| 7    | 12353        | 1              |
| 8    | 12354        | 1              |
| 9    | 12355        | 1              |
| 10   | 12356        | 3              |
| 11   | 12357        | 1              |
|      |              | _              |

#### • 각 고객 별로 구매한 아이템의 총 수량 더하기

SELECT
CustomerID,
SUM(Quantity) AS item\_cnt
FROM
velvety-study-473605-t0.modulabs\_project.data
WHERE
Quantity > 0
GROUP BY
CustomerID;

| 행 // | CustomerID | · //  | item_cnt ▼ |
|------|------------|-------|------------|
| 1    |            | 12346 | 74215      |
| 2    |            | 12347 | 2458       |
| 3    |            | 12348 | 2332       |
| 4    |            | 12349 | 630        |
| 5    |            | 12350 | 196        |
| 6    |            | 12352 | 526        |
| 7    |            | 12353 | 20         |
| 8    |            | 12354 | 530        |
| 9    |            | 12355 | 240        |

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

CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs\_project.user\_rf AS
WITH purchase\_cnt AS (
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase\_cnt
FROM
velvety-study-473605-t0.modulabs\_project.data
GROUP BY

```
CustomerID
),
item_cnt AS (
 SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
 FROM
 velvety-study-473605-t0.modulabs_project.data
 WHERE
 Quantity > 0
 GROUP BY
  CustomerID
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 velvety-study-473605-t0.modulabs_project.user_r AS ur
 ON pc.CustomerID = ur.CustomerID;
FROM velvety-study-473605-t0.modulabs_project.user_rf
```

| 행 | CustomerID ▼ | purchase_cnt ▼ // | item_cnt ▼ | Recency ▼ ↓ |
|---|--------------|-------------------|------------|-------------|
| 1 | 15165        | 1                 | 160        | 373         |
| 2 | 13747        | 1                 | 8          | 373         |
| 3 | 16583        | 1                 | 111        | 373         |
| 4 | 17908        | 1                 | 169        | 373         |
| 5 | 13065        | 1                 | 74         | 373         |
| 6 | 17643        | 1                 | 71         | 373         |
| 7 | 14237        | 1                 | 38         | 373         |
| 8 | 12791        | 1                 | 96         | 373         |
| 9 | 18074        | 1                 | 190        | 373         |

#### **Monetary**

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice), 0) AS user_total
FROM
velvety-study-473605-t0.modulabs_project.data
GROUP BY
CustomerID
HAVING
SUM(Quantity * UnitPrice) > 0;
```

#### 쿼리 결과

| 작업 정 | 보 결과         | 시각화   | JSON    | 실행 세- |
|------|--------------|-------|---------|-------|
| 행 // | CustomerID ▼ | user_ | total ▼ |       |
| 1    | 1234         | 47    | 4310.0  |       |
| 2    | 1234         | 48    | 1437.0  |       |
| 3    | 123          | 49    | 1458.0  |       |
| 4    | 123          | 50    | 294.0   |       |
| 5    | 123          | 52    | 1265.0  |       |
| 6    | 123          | 53    | 89.0    |       |
| 7    | 123          | 54    | 1079.0  |       |
| 8    | 123          | 55    | 459.0   |       |
| 9    | 123          | 56    | 2487.0  |       |
| 10   | 123          | 57    | 6208 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 velvety-study-473605-t0.modulabs_project.user_rfm AS
SELECT
 rf.CustomerID,
 rf.purchase_cnt,
 rf.item_cnt,
 rf.Recency,
 ut.user_total,
 ROUND(ut.user_total / rf.purchase_cnt, 2) AS user_average
 velvety-study-473605-t0.modulabs_project.user_rf AS rf
LEFT JOIN (
 SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 0) AS user_total
 velvety-study-473605-t0.modulabs_project.data
 GROUP BY
  CustomerID
 HAVING
  SUM(Quantity * UnitPrice) > 0
) AS ut
ON
 rf.CustomerID = ut.CustomerID;
```

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user\_rfm인 새 테이블이 생성되었습니다.

# RFM 통합 테이블 출력하기

• 최종 user\_rfm 테이블을 출력하기

SELECT \*
FROM velvety-study-473605-t0.modulabs\_project.user\_rfm
LIMIT 10;



# 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 velvety-study-473605-t0.modulabs_project.user_data AS
WITH
unique_products AS (
 SELECT
  COUNT(DISTINCT StockCode) AS unique_products
  velvety-study-473605-t0.modulabs_project.data
 GROUP BY
  CustomerID
SELECT
up.unique_products
FROM
velvety-study-473605-t0.modulabs_project.user_rfm AS ur
unique_products AS up
ON ur.CustomerID = up.CustomerID;
SELECT *
FROM velvety-study-473605-t0.modulabs_project.user_data
```



# 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 velvety-study-473605-t0.modulabs_project.user_data AS
WITH
purchase_intervals AS (
 SELECT
  CustomerID,
  COALESCE(ROUND(AVG(interval_days), 2), 0) AS average_interval
 FROM (
  SELECT
   CustomerID,
    DATE_DIFF(
     DATE(InvoiceDate),
     LAG(DATE(InvoiceDate)) OVER (PARTITION BY CustomerID ORDER BY DATE(InvoiceDate)),
   ) AS interval_days
  FROM
   velvety-study-473605-t0.modulabs_project.data
 GROUP BY
   CustomerID
SELECT
u.* EXCEPT (average_interval),
pi.average_interval
FROM
velvety-study-473605-t0.modulabs_project.user_data AS u
LEFT JOIN
purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
SELECT *
FROM velvety-study-473605-t0.modulabs_project.user_data
```

| 작업 정 | 보 결과         | 시각화 JSON       | 실행 세부정보      | 실행 그래프    |              |                   |                   |                    |
|------|--------------|----------------|--------------|-----------|--------------|-------------------|-------------------|--------------------|
| 1    | CustomerID ▼ | purchase_cnt - | , item_cnt ▼ | Recency ▼ | user_total ▼ | user_average ▼ // | unique_products 🕶 | average_interval 🕶 |
| 1    | 16881        | 1              | 600          | 66        | 432.0        | 432.0             | 1                 | 0.0                |
| 2    | 13302        | 1              | 5            | 155       | 64.0         | 64.0              | 1                 | 0.0                |
| 3    | 16344        | 1              | 18           | 158       | 101.0        | 101.0             | 1                 | 0.0                |
| 4    | 17763        | 1              | 12           | 263       | 15.0         | 15.0              | 1                 | 0.0                |
| 5    | 16737        | 1              | 288          | 53        | 418.0        | 418.0             | 1                 | 0.0                |
| 6    | 15195        | 1              | 1404         | 2         | 3861.0       | 3861.0            | 1                 | 0.0                |
| 7    | 16990        | 1              | 100          | 218       | 179.0        | 179.0             | 1                 | 0.0                |
| 8    | 13307        | 1              | 4            | 120       | 15.0         | 15.0              | 1                 | 0.0                |
| 9    | 14424        | 1              | 48           | 17        | 322.0        | 322.0             | 1                 | 0.0                |
| 10   | 13135        | 1              | 4300         | 196       | 3096.0       | 3096.0            | 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

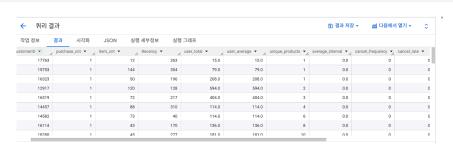
 취소 빈도와 취소 비율을 계산하고 그 결과를 user\_data 에 통합하기 (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs_project.user_data AS
WITH
TransactionInfo AS (
 SELECT
  CustomerID,
   COUNT(DISTINCT InvoiceNo) AS total_transactions,
   DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END
  ) AS cancel_frequency
  FROM
  velvety-study-473605-t0.modulabs_project.data
  GROUP BY
   CustomerID
SELECT
u.CustomerID,
u.purchase_cnt,
u.item_cnt,
u.Recency,
u.user_total,
u.user_average,
u.unique_products,
u.average_interval,
COALESCE(t.cancel_frequency, 0) AS cancel_frequency,
COALESCE(
 ROUND((t.cancel_frequency / t.total_transactions) * 100, 2),
) AS cancel_rate
FROM
velvety-study-473605-t0.modulabs_project.user_data AS u
LEFT JOIN
TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

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 velvety-study-473605-t0.modulabs\_project.user\_data



# 회고

Keep : 중간중간 잘 돌아가는지 확인하여서 오류가 생기면 그때그때 확인하고 고칠 수 있었던 점이 좋았습니다. 또한 흐름에 맞춰 코드 중간중간 구간을 표시하여 코드를 되돌아가야하는 때가 생겼을때 빠르게 다시 찾을 수 있었습니다. 그 점이 좋았습니다.

Problem :쿼리 결과에서 페이지를 넘길 수 있다는 사실을 몰라서

답지와 고객 수, recency값이 다르게 나온것처럼 보여(행이 순서대로 나오지 않는다는 점을 간과하였음) 원인을 찾는 데 많은 시간이 소요되었습니다..

Try : 다음 프로젝트에서는 분석 목표에 따른 데이터 정제 규칙을 먼저 명확히 정의하여, 정답지와의 결과 차이로 인한 혼란을 줄이겠습니다.