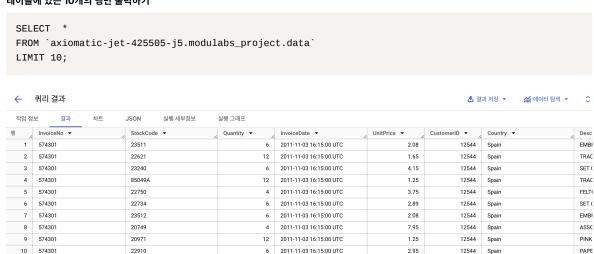
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

데이터 수 세기

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

```
SELECT

COUNT(InvoiceNo) AS InvoiceNo_count,

COUNT(StockCode) AS StockCode_count,

COUNT(Description) AS Description_count,

COUNT(Quantity) AS Quantity_count,

COUNT(InvoiceDate) AS InvoiceDate_count,

COUNT(UnitPrice) AS UnitPrice_count,

COUNT(CustomerID) AS CustomerID_count,
```



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_perc
FROM `axiomatic-jet-425505-j5.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_perc
FROM `axiomatic-jet-425505-j5.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_pe
FROM `axiomatic-jet-425505-j5.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_perce
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
SELECT 'InvoiceDate' AS column_name,
      ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_pe
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
UNTON ALL
SELECT 'UnitPrice' AS column name,
      ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perc
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
SELECT 'CustomerID' AS column_name,
      ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_per
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
SELECT 'Country' AS column_name,
       ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percen
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```

← 쿼리 결과

작업 정	보 결과 차트	JSON 실행 세탁	부정보 실행 그래프
행 //	column_name ▼	missing_percentage	
1	Country	0.0	
2	UnitPrice	0.0	
3	CustomerID	24.93	
4	Description	0.27	
5	InvoiceDate	0.0	
6	Quantity	0.0	
7	InvoiceNo	0.0	
8	StockCode	0.0	

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
WHERE StockCode = '85123A';
```



결측치 처리

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

```
DELETE FROM `axiomatic-jet-425505-j5.modulabs_project.data1`
WHERE InvoiceNo IS NULL
OR StockCode IS NULL
OR Description IS NULL
OR Quantity IS NULL
OR InvoiceDate IS NULL
OR UnitPrice IS NULL
OR CustomerID IS NULL
OR Country 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

중복값 처리

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

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기



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



-1 2010-12-21 12:33:00 UTC

-2 2010-12-21 12:33:00 UTC

2010-12-21 12:33:00 UTC

1.25

4.95

10.75

18176 Unite

18176 Unite

18176 Unite

HANGING HEART JAR T-LIGHT ...

RETROSPOT HEART HOT WAT...

BROCANTE SHELF WITH HOOKS

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

84978

21485

22832

C539709

8 C539709

```
ROUND((SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 1) AS cancelled
```



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

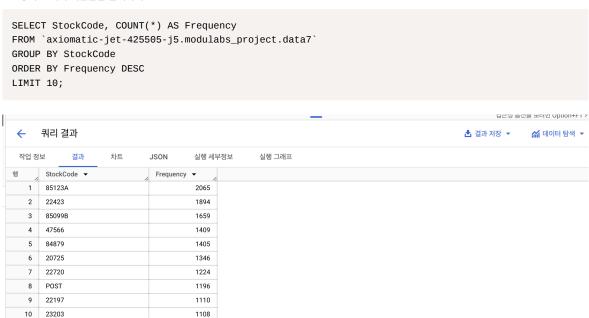
```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `axiomatic-jet-425505-j5.modulabs_project.data1`;

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

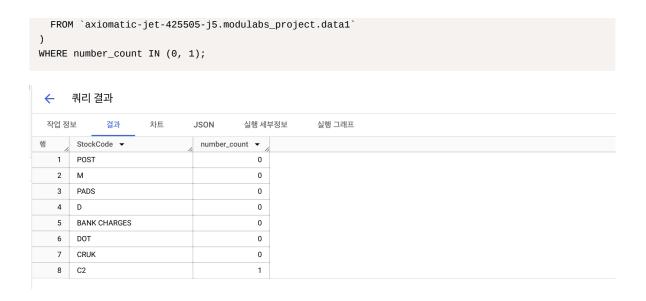
행 unique_stockcode_cg
1 3684
```

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



- 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
```



- 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, '[0-9]', '')) AS number_count
 FROM `axiomatic-jet-425505-j5.modulabs_project.data1`
WHERE number_count IN (0, 1);
WITH FilteredData AS (
 SELECT StockCode
 FROM (
   SELECT StockCode,
     LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
   FROM `axiomatic-jet-425505-j5.modulabs_project.data`
 WHERE number_count IN (0, 1)
SELECT
 ROUND((COUNT(*) / (SELECT COUNT(*) FROM `axiomatic-jet-425505-j5.modulabs_project.data`)) * 100, 2
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
WHERE StockCode IN (SELECT StockCode FROM FilteredData);
```

← 쿼리 결과

작업 정!	보 결과	차트	JSON	실행 세부정보	실행 그래프
행	행 percentage ▼				
1	0.4	8			

• 제품과 관련되지 않은 거래 기록을 제거하기

```
-- 숫자가 0~1인 값을 가진 StockCode를 추출
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.filtered_stockcodes` AS
SELECT StockCode
```

```
FROM (
 SELECT StockCode,
   LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
 FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
WHERE number_count IN (0, 1);
-- 해당 StockCode를 포함한 기록을 제외하고 데이터를 다시 테이블에 저장
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.data7_filtered` AS
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
WHERE StockCode NOT IN (SELECT StockCode FROM `axiomatic-jet-425505-j5.modulabs_project.filtered_sto
-- 전체 행 개수
SELECT COUNT(*) AS total_count_before
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`;
-- 숫자가 0~1인 값을 가진 StockCode를 추출
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.filtered_stockcodes` AS
SELECT StockCode
FROM (
 SELECT StockCode,
   LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
 FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
WHERE number count IN (0, 1):
-- 필터링된 데이터로 새로운 테이블 생성
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.data7_filtered` AS
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
WHERE StockCode NOT IN (SELECT StockCode FROM `axiomatic-jet-425505-j5.modulabs_project.filtered_sto
-- 필터링 후의 전체 행 개수
SELECT COUNT(*) AS total_count_after
FROM `axiomatic-jet-425505-j5.modulabs_project.data7_filtered`;
-- 삭제된 행 개수 계산
SELECT
  (SELECT COUNT(*) FROM `axiomatic-jet-425505-j5.modulabs_project.data7`) -
  (SELECT COUNT(*) FROM `axiomatic-jet-425505-j5.modulabs_project.data7_filtered`) AS deleted_count;
    쿼리 결과
 작업 정보
           결과
                  차트
                         JSON
                                  실행 세부정보
                                              실행 그래프
      deleted_count ▼
             1915
```

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS Frequency
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
GROUP BY Description
ORDER BY Frequency DESC
LIMIT 30;
```



• 서비스 관련 정보를 포함하는 행들을 제거하기



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.data7_uppercase` AS
SELECT
 InvoiceNo,
 StockCode,
 UPPER(Description) AS Description,
  Quantity,
 InvoiceDate,
 UnitPrice,
 CustomerID,
 Country
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`;
#2
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.data7_uppercase AS
SELECT
  * EXCEPT (Description),
 UPPER(Description) AS Description
FROM axiomatic-jet-425505-j5.modulabs_project.data7;
 ← 쿼리 결과
                                                                                               占 결과 저장
 작업 정보
                 실행 세부정보
                            실행 그래프
          결과
 이 문으로 이름이 data7_uppercase인 새 테이블이 생성되었습니다.
```



UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기



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





• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.data7_consistent AS SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
WHERE UnitPrice != 0;
```



작업 정보 <u>결과</u> 실행 세부정보 실행 그래프

① 이 문으로 이름이 data7_consistent인 새 테이블이 생성되었습니다.

11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기



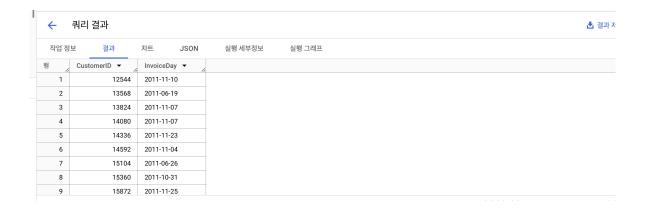


• 가장 최근 구매 일자를 MAX() 함수로 찾아보기



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
MAX(PARSE_DATE('%Y-%m-%d', FORMAT_DATE('%Y-%m-%d', CAST(InvoiceDate AS DATE)))) AS InvoiceDay
FROM `axiomatic-jet-425505-j5.modulabs_project.data5`
GROUP BY CustomerID;
```



• 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산하기

```
SELECT
  CustomerID,
  DATE_DIFF(MAX(InvoiceDay) OVER (), InvoiceDay, DAY) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  GROUP BY CustomerID
);
```



• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user_r 이라는 이름의 테이블로 저장하기

```
# InvoiceDate 컬럼을 연월일 자료형으로 변경하여 새로운 테이블 생성

CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.data7_cleaned AS

SELECT

PARSE_DATE('%Y-%m-%d', FORMAT_DATE('%Y-%m-%d', CAST(InvoiceDate AS DATE))) AS InvoiceDay,
InvoiceNo,
StockCode,
UPPER(Description) AS Description,
Quantity,
UnitPrice,
CustomerID,
Country

FROM
axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
WHERE
```

```
UnitPrice != 0;
# 가장 최근 구매 일자 찾기
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.most_recent_purchase_date AS
SELECT
 MAX(InvoiceDay) AS most_recent_purchase_date
FROM
 axiomatic-jet-425505-j5.modulabs_project.data7_cleaned;
# 유저별 가장 큰 InvoiceDay 찾기
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.customer_last_purchase AS
SELECT
 CustomerID,
 MAX(InvoiceDay) AS InvoiceDay
 axiomatic-jet-425505-j5.modulabs_project.data7_cleaned
GROUP BY
 CustomerID;
# 가장 최근 일자와 유저별 마지막 구매일 간 차이 계산하여 최종 데이터셋 생성
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.user_r AS
 clp.CustomerID,
 DATE_DIFF(mrd.most_recent_purchase_date, clp.InvoiceDay, DAY) AS recency
FROM
  axiomatic\text{-}jet\text{-}425505\text{-}j5.modulabs\_project.customer\_last\_purchase clp,\\
  axiomatic-jet-425505-j5.modulabs_project.most_recent_purchase_date mrd;
# 완성 테이블 조회
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_r;
← 쿼리 결과
 작업 정보
                  실행 세부정보
                               실행 그래프
    이 문으로 이름이 user_r인 테이블이 교체되었습니다.
     쿼리 결과
  \leftarrow
  작업 정보
                           JSON
                                    실행 세부정보
                                                실행 그래프
       CustomerID ▼
                              0
    2
              17428
                              0
    3
              13069
                              0
    4
              16954
                              0
    5
              14446
                              0
    6
              13113
                              0
    7
              17315
                              0
    8
              17490
                              0
    9
              15498
                              0
```

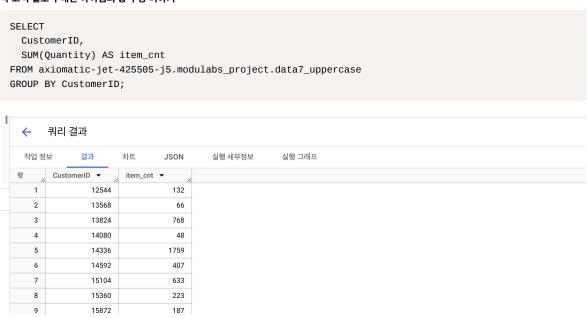
Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
```



• 각 고객 별로 구매한 아이템의 총 수량 더하기



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

```
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT InvoiceNo) AS purchase_cnt
    FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
    GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
    SELECT
```

```
CustomerID,
    SUM(Quantity) AS item_cnt
  FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  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-jet-425505-j5.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
#테이블 조회
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_rf;
      쿼리 결과
  작업 정보
            결과
                    실행 세부정보
                                 실행 그래프
      이 문으로 이름이 user_rf인 테이블이 교체되었습니다.
   쿼리 결과
                                                                                                      占 결과
 작업 정보
                          JSON
                                  실행 세부정보
                                               실행 그래프
      CustomerID ▼
             12713
                                        508
                                                      0
   2
             12792
                             1
                                        217
                                                     256
   3
             15083
                                         38
                                                     256
             18010
                                         60
                                                     256
   5
             13436
                                         76
   6
             15520
                                        314
   7
             14569
                                         79
   8
             13298
                                         96
             14476
                                        110
                                                     257
```

Monetary

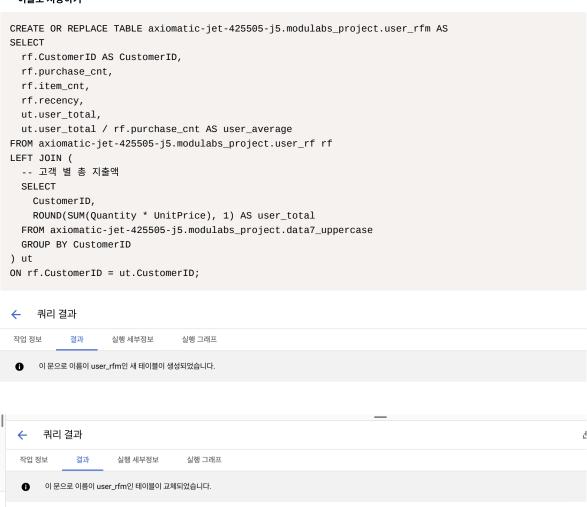
• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
GROUP BY CustomerID;
```



• 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



RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

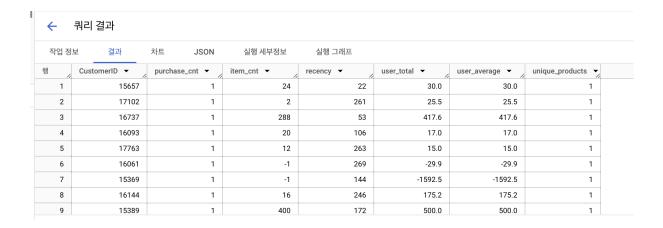


11-8. 추가 Feature 추출

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



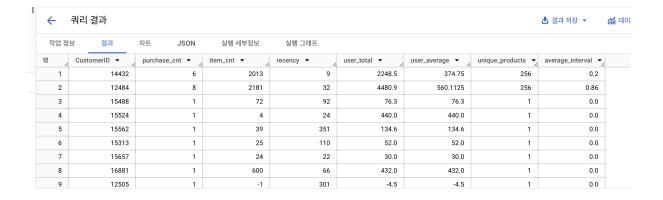


2. 평균 구매 주기

- 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 axiomatic-jet-425505-j5.modulabs_project.user_data AS
WITH purchase_intervals AS (
  -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
  SELECT
   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)
     axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
   WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM axiomatic-jet-425505-j5.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
# 결과 테이블 조회
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_data;
```

수 쿼리 결과 작업 정보 결과 실행 시부정보 실행 그래프 ① 모으로 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 axiomatic-jet-425505-j5.modulabs_project.user_data AS
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 END) AS cancel_frequency
  {\tt FROM}\ axiomatic\text{-}{\tt jet\text{-}425505\text{-}{\tt j5}.modulabs\_project.data7\_uppercase}
  GROUP BY CustomerID
)
SELECT u.*,
        t.total_transactions,
        t.cancel_frequency,
       ROUND(CAST(t.cancel\_frequency\ AS\ FLOAT64)\ /\ t.total\_transactions,\ 2)\ AS\ cancel\_rate
FROM axiomatic-jet-425505-j5.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```



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

```
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_data;
```

