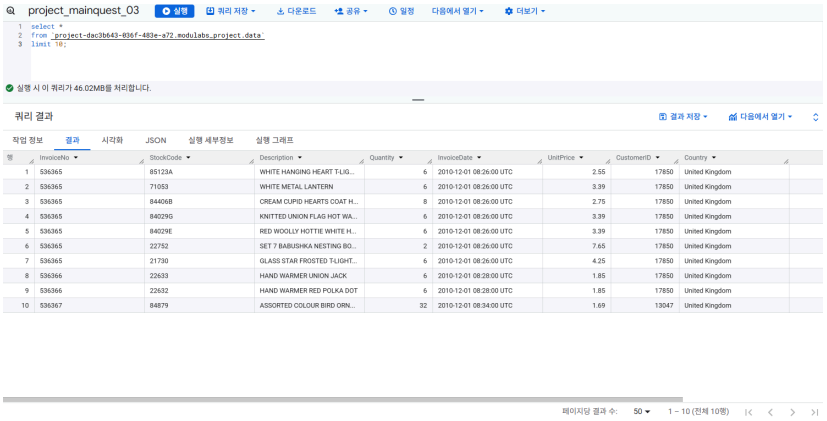
고객을 세그먼테이션하자 [프로젝트] - 김수경

11-2. 데이터 불러오기

데이터 살펴보기

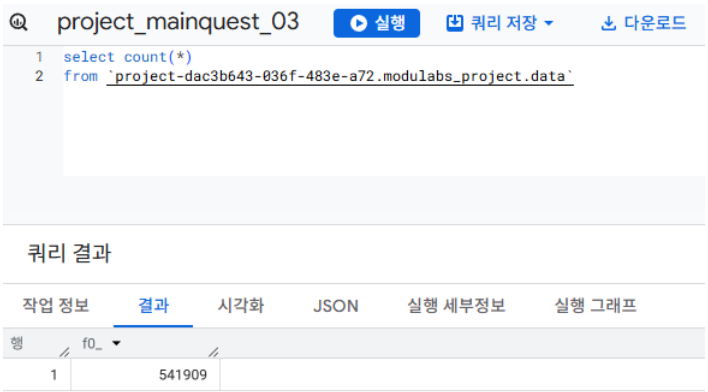
- **테이블에 있는 10개의 행만 출력하기**

```
select *
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
limit 10;
```



- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
select count(*)
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

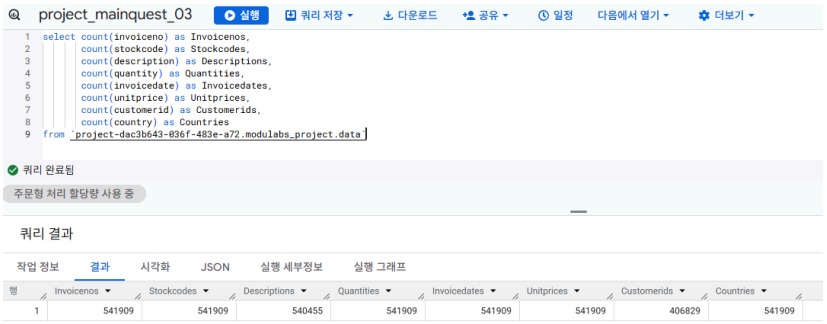


  - 541,909행

데이터 수 세기

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

```
select count(invoiceno) as Invoicenos,
      count(stockcode) as Stockcodes,
      count(description) as Descriptions,
      count(quantity) as Quantities,
      count(invoicedate) as Invoicedates,
      count(unitprice) as Unitprices,
      count(customerid) as Customerids,
      count(country) as Countries
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```



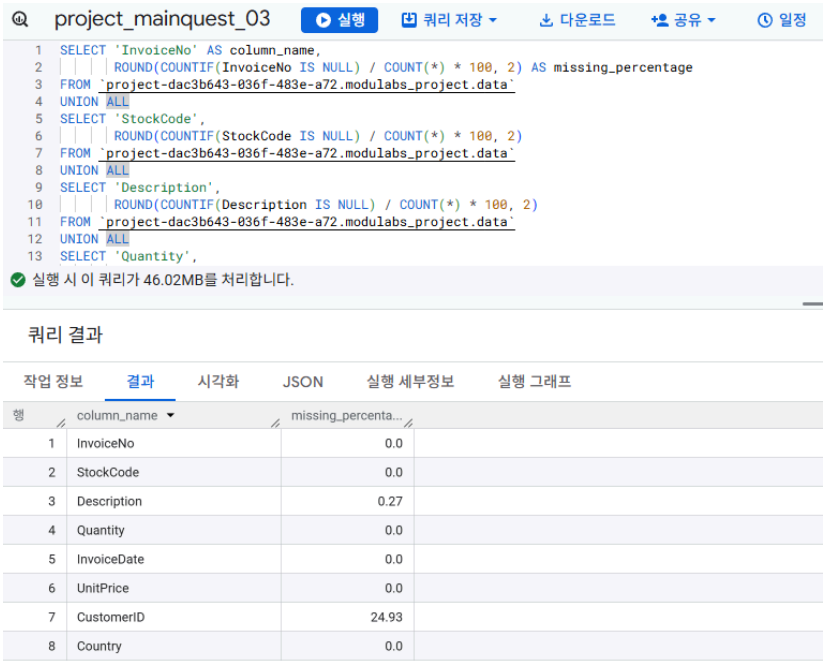
11-4. 데이터 전처리 방법(1): 결측치 제거

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(COUNTIF(InvoiceNo IS NULL) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'StockCode',
      ROUND(COUNTIF(StockCode IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'Description',
      ROUND(COUNTIF(Description IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'Quantity',
      ROUND(COUNTIF(Quantity IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'InvoiceDate',
      ROUND(COUNTIF(InvoiceDate IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'UnitPrice',
      ROUND(COUNTIF(UnitPrice IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'CustomerID',
      ROUND(COUNTIF(CustomerID IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'Country',
```

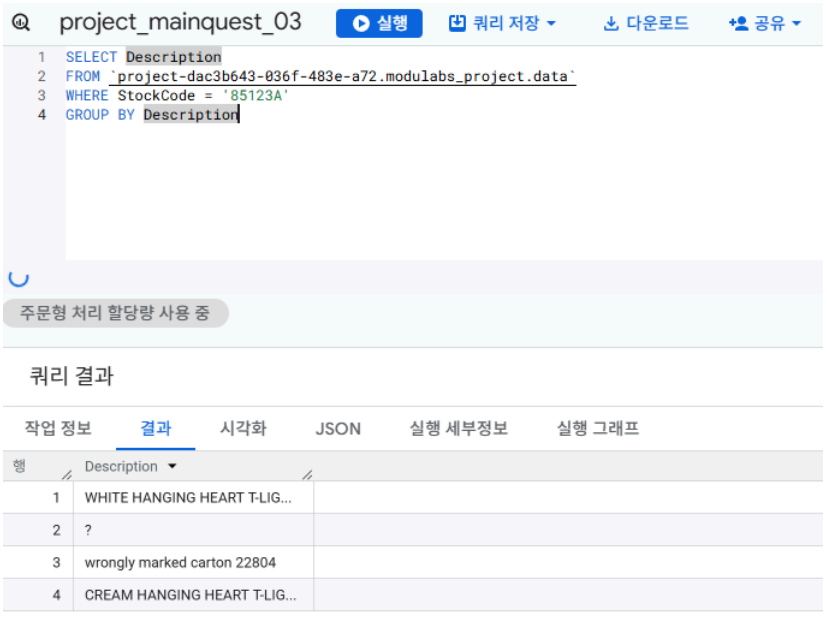
```
    ROUND(COUNTIF(Country IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;
```



결측치 처리 전략

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

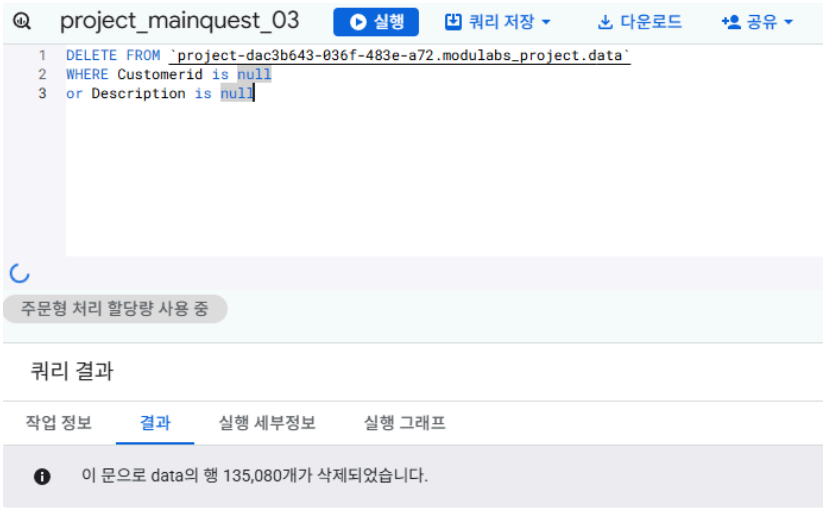
```
SELECT Description
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE StockCode = '85123A'
GROUP BY Description
```



결측치 처리 (⭐delete 후 이전 원본 데이터를 재활용하기)

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

```
DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE Customerid is null
or Description is null
```

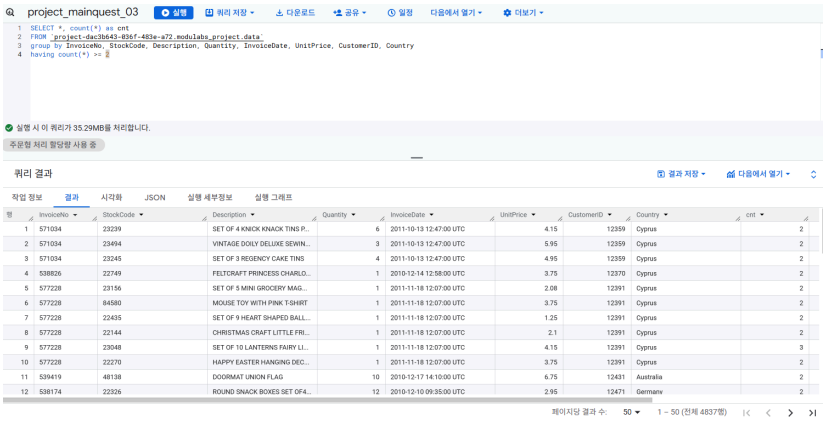


11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- **중복된 행의 수를 세어보기**
  - **8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기**

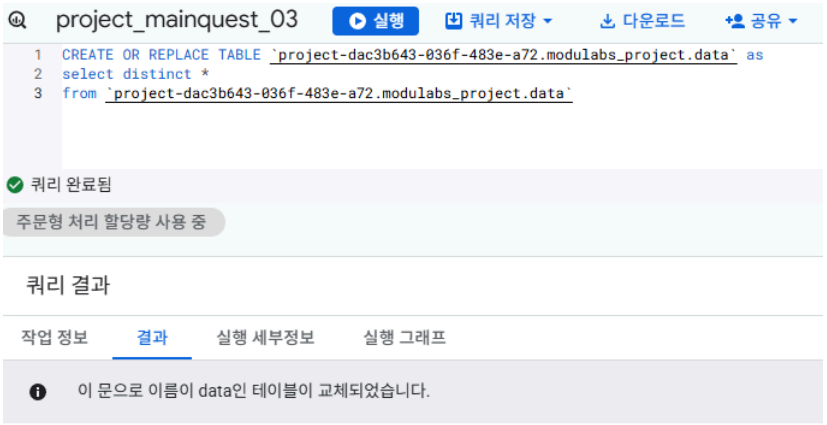
```
SELECT *, count(*) as cnt
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
having count(*) >= 2
```



중복값 처리

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

```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` as
select distinct *
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```



11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
select count(invoiceno)
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by invoiceno
```

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
select invoiceno
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

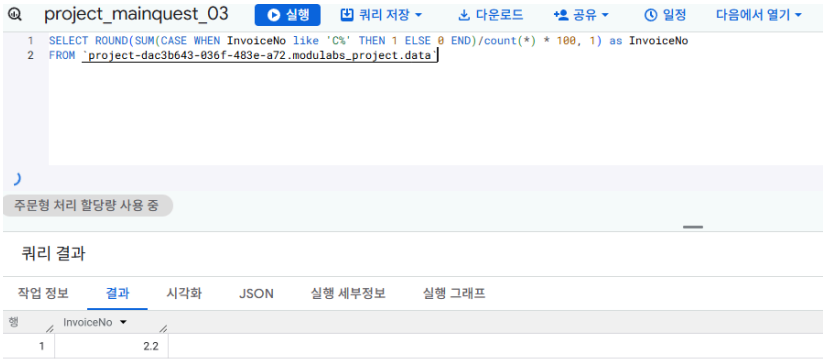
```
group by invoiceno
order by invoiceno
limit 100
```

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

```
select invoiceno
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
where invoiceno like 'C%'
group by invoiceno
order by invoiceno
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)/count(*) * 100, 1) as InvoiceNo
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
```



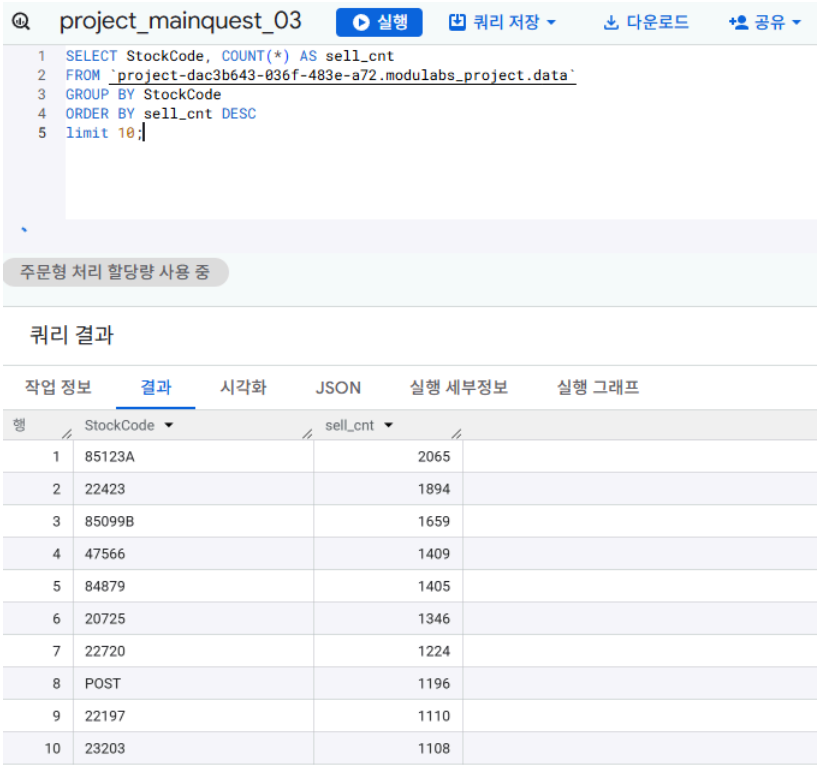
StockCode 살펴보기

- **고유한 StockCode의 개수를 출력하기**

```
SELECT count(stockcode)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
GROUP BY stockcode
```

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

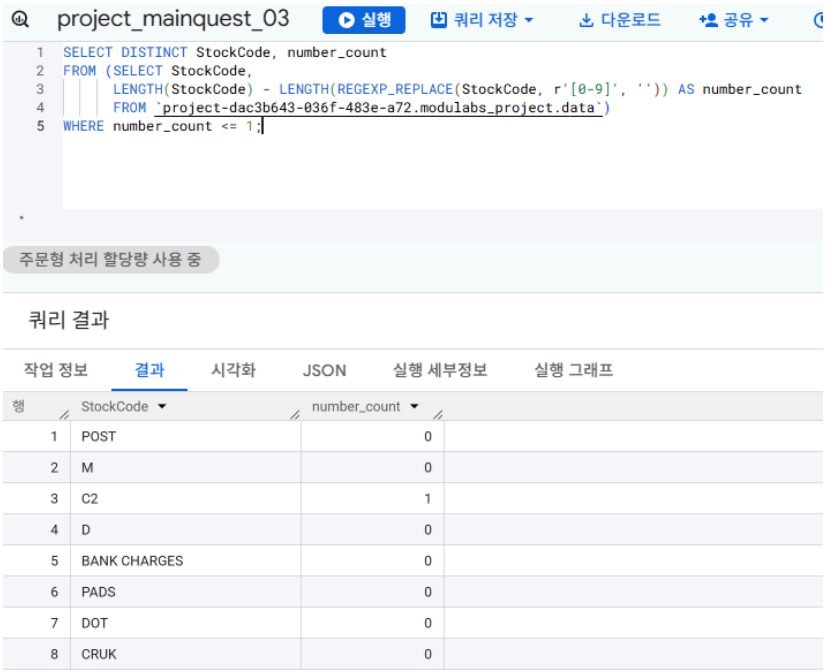
```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
limit 10;
```



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 `project-dac3b643-036f-483e-a72.modulabs_project.data`)
WHERE number_count <= 1;
```



행	StockCode	number_count
1	POST	0
2	M	0
3	C2	1
4	D	0
5	BANK CHARGES	0
6	PADS	0
7	DOT	0
8	CRUK	0

- `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)

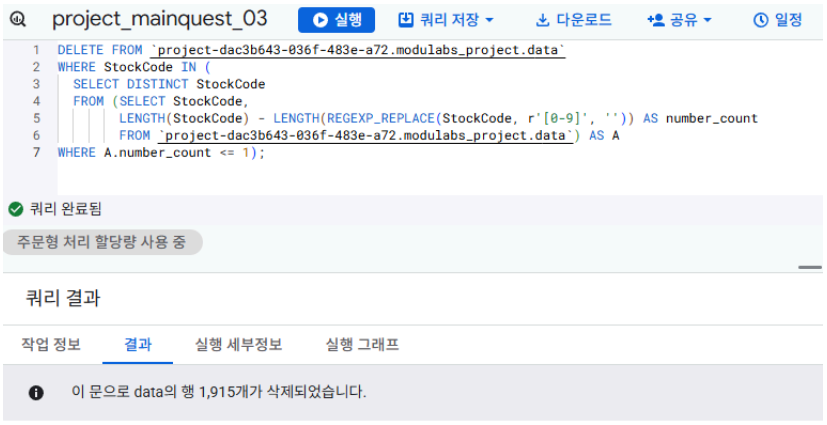
```
WITH stockcode_digits as (SELECT StockCode,
                LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
                FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`)
SELECT ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END)/count(*) * 100, 2) as StockCode_per
FROM stockcode_digits
```

[결과 이미지를 넣어주세요]

  - (이미 DELETE 함수를 사용해 이미지 첨부 불가..)
    - 답 : 0.48%

- **제품과 관련되지 않은 거래 기록을 제거하기**

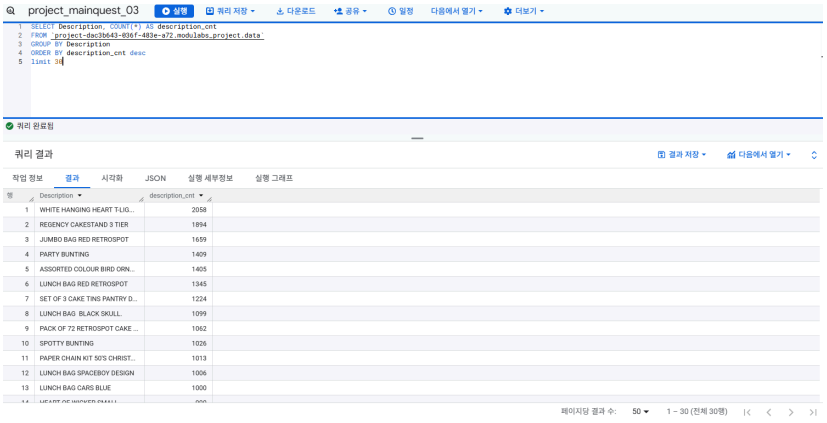
```
DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE StockCode IN (
 SELECT DISTINCT StockCode
 FROM (SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
      FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`) AS A
WHERE A.number_count <= 1);
```



Description 살펴보기

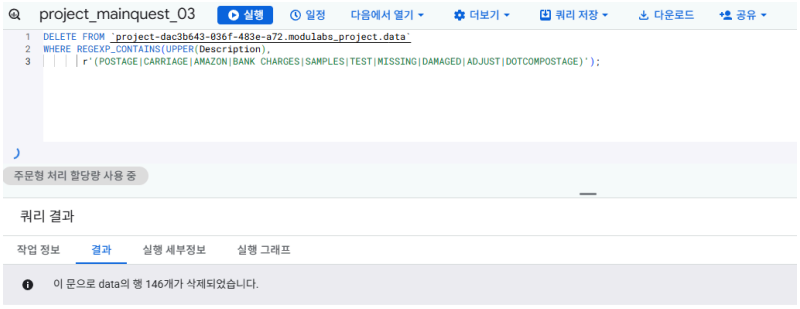
- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS description_cnt
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt desc
limit 30
```



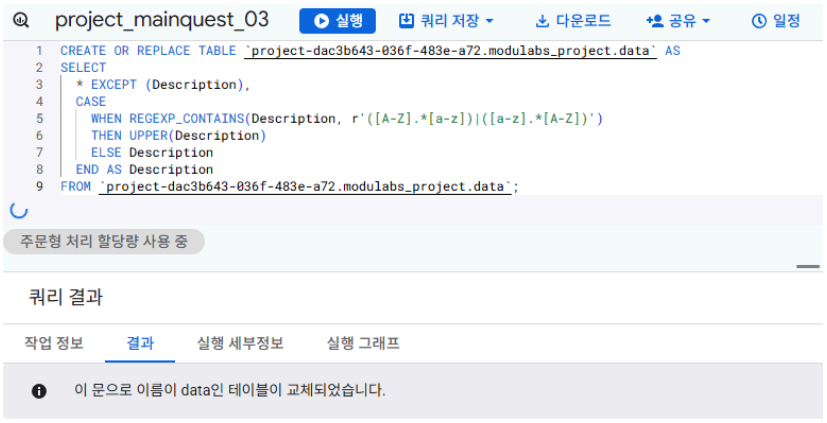
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE REGEXP_CONTAINS(UPPER(Description),
    r'(POSTAGE|CARRIAGE|AMAZON|BANK CHARGES|SAMPLES|TEST|MISSING|DAMAGED|ADJUST|DOTCOMPOSTAGE)');
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

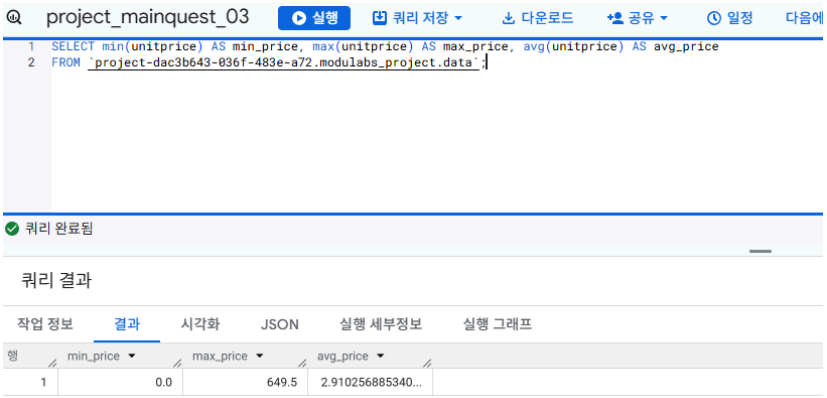
```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  CASE
    WHEN REGEXP_CONTAINS(Description, r'([A-Z].*[a-z])|([a-z].*[A-Z])')
    THEN UPPER(Description)
    ELSE Description
  END AS Description
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;
```



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT min(unitprice) AS min_price, max(unitprice) AS max_price, avg(unitprice) AS avg_price
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;
```



- **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 `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE unitprice = 0
```

- **UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기**

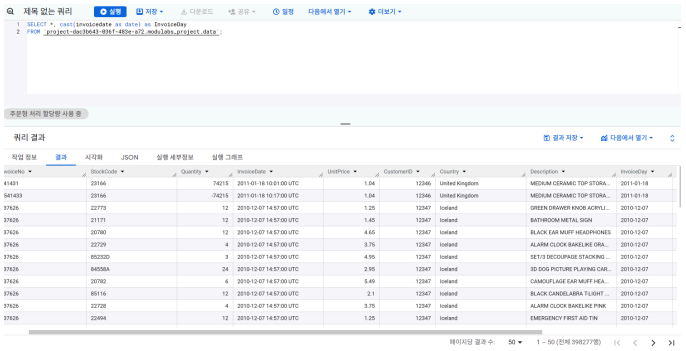
```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` AS
SELECT *
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE unitprice != 0
```

11-7. RFM 스코어

Recency

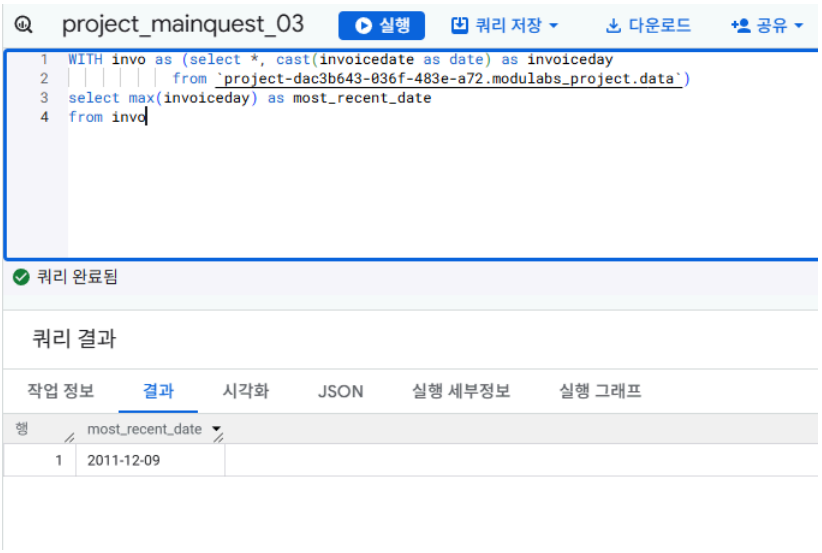
- `InvoiceDate` **컬럼을 연월일 자료형으로 변경하기**

```
SELECT *, cast(invoicedate as date) as InvoiceDay
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;
```



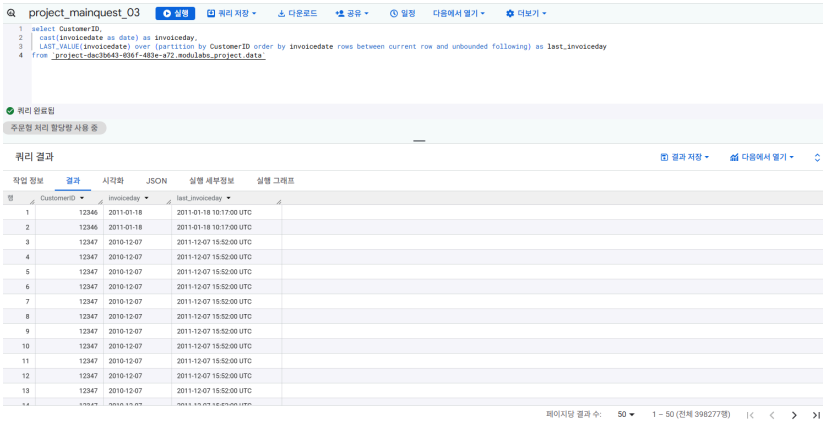
- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
WITH invo as (select *, cast(invoicedate as date) as invoiceday
        from `project-dac3b643-036f-483e-a72.modulabs_project.data`)
select max(invoiceday) as most_recent_date
from invo
```



- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

```
select CustomerID,
  cast(invoicedate as date) as invoiceday,
  LAST_VALUE(invoicedate) over (partition by CustomerID order by invoicedate rows between current row and unbounded fol
lowing) as last_invoiceday
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```



- **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
);
```

제목 없는 쿼리 실행 저장 다운로드 공유

```sql
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
  GROUP BY CustomerID
);
```

실행 시 이 쿼리가 6.08MB를 처리합니다.

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	CustomerID	recency
1	12354	232
2	12388	15
3	12452	16
4	12633	58
5	12688	113
6	12830	37
7	12849	31
8	13426	0
9	13492	136
10	13507	3
11	13727	28
12	13953	7
13	14218	42

- **최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 `user_r` 이라는 이름의 테이블로 저장하기**

```sql
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_r` AS
WITH user_r as (SELECT CustomerID,
            EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
            FROM (SELECT CustomerID,
                MAX(DATE(InvoiceDate)) AS InvoiceDay
                FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
                GROUP BY CustomerID))
select *
from user_r;
```

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
select customerID, count(distinct invoiceno) as purchase_cnt
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by customerID
```

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
select customerID, sum(Quantity) as item_cnt
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by customerID
```

제목 없는 쿼리 실행 저장 다운로드 공유

```
1  select customerID, sum(Quantity) as item_cnt
2  from `project-dac3b643-036f-483e-a72.modulabs_project.data`
3  group by customerID
```

쿼리 완료됨

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	customerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240
10	12356	1573
11	12357	2708
12	12358	242
13	12359	1599
14	12360	1156
15	12361	90
16	12362	2168
17	12363	408

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

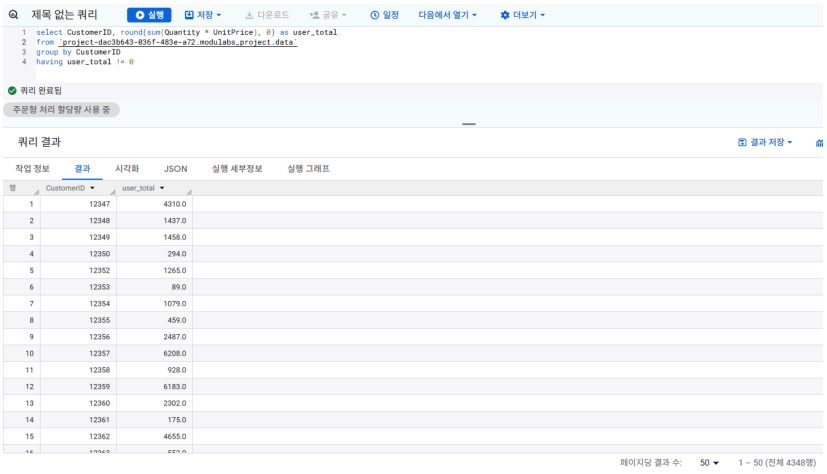
```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
select customerID, count(distinct invoiceno) as purchase_cnt
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by customerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
select customerID, sum(Quantity) as item_cnt
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
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 `project-dac3b643-036f-483e-a72.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

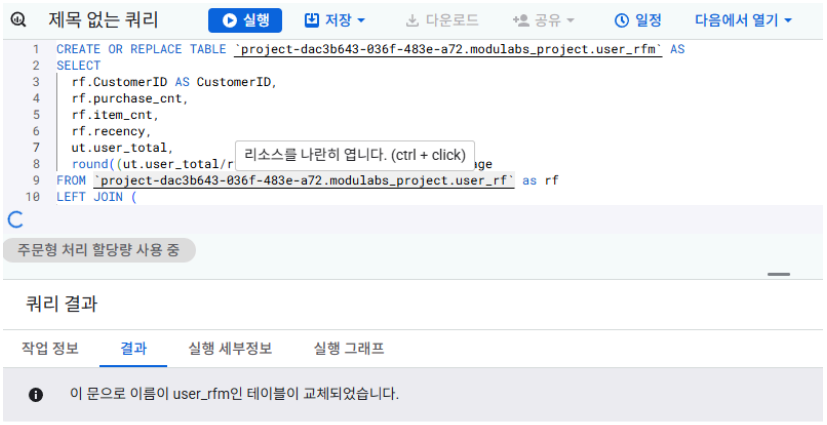
```
select CustomerID, round(sum(Quantity * UnitPrice), 0) as user_total
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by CustomerID
having user_total != 0
```



- **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 `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm` AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
```

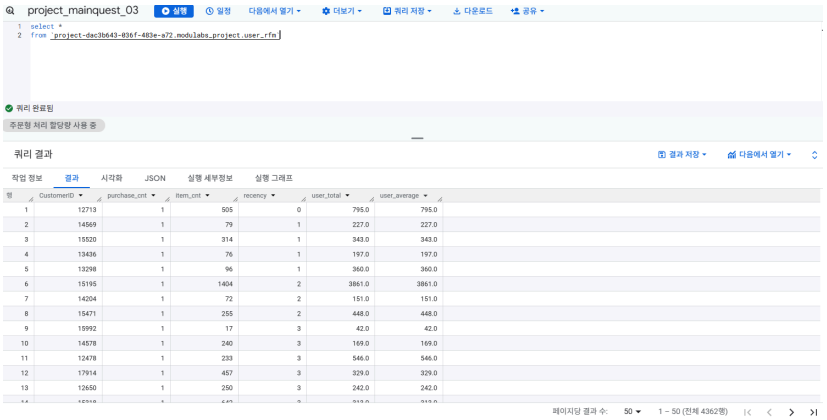
```
  rf.recency,
  ut.user_total,
  round((ut.user_total/rf.purchase_cnt), 0) as user_average
FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_rf` as rf
LEFT JOIN (
  select CustomerID, round(sum(Quantity * UnitPrice), 0) as user_total
    from `project-dac3b643-036f-483e-a72.modulabs_project.data`
    group by CustomerID
    having user_total != 0) as ut
ON rf.CustomerID = ut.CustomerID;
```



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select *
from `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm`
```



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**

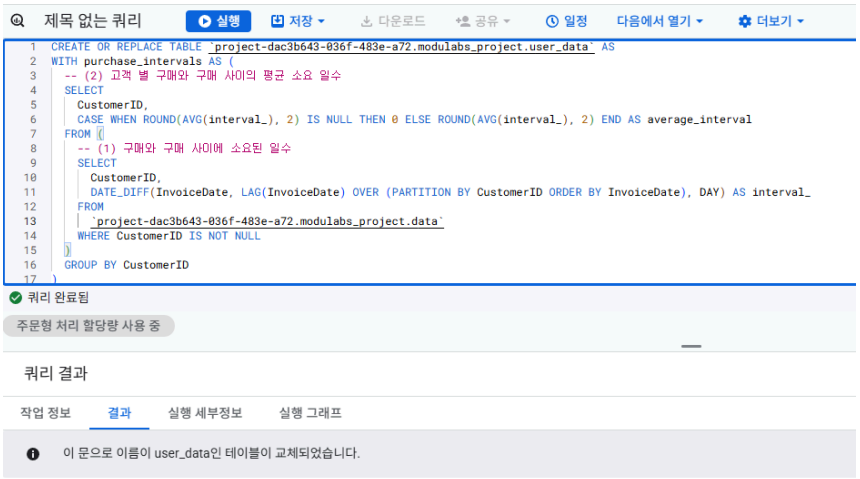
```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

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-dac3b643-036f-483e-a72.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
 FROM (
   -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
   FROM
     `project-dac3b643-036f-483e-a72.modulabs_project.data`
   WHERE CustomerID IS NOT NULL
 )
```

```
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```



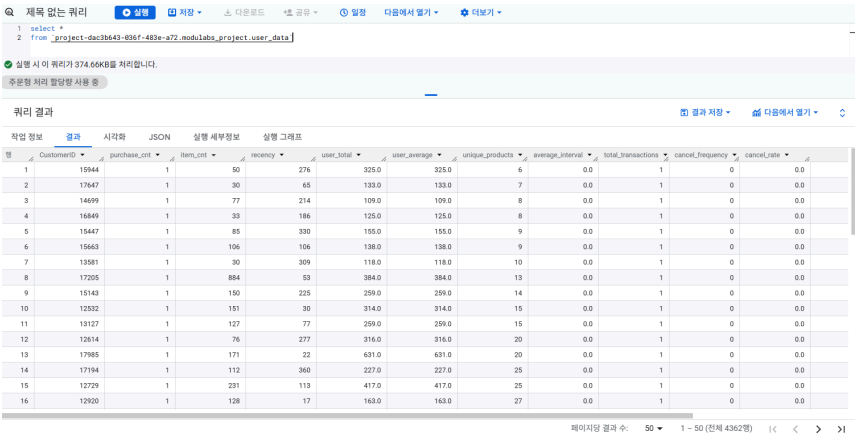
3. 구매 취소 경향성

- **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-dac3b643-036f-483e-a72.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT invoiceno) AS total_transactions,
    COUNT(DISTINCT CASE WHEN invoiceno LIKE 'C%' THEN invoiceno END) AS cancel_frequency
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
  WHERE customerID IS NOT NULL
  GROUP BY customerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency * 1.0 / t.total_transactions, 2) as cancel_rate
FROM `project-dac3b643-036f-483e-a72.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 `project-dac3b643-036f-483e-a72.modulabs_project.user_data`
```



회고

Keep : 최대한 직접 코드를 작성해보며 화면과 다른 값이 나왔을 때 직접 오류를 수정해보는 과정을 겪어보며 어떤 코드를 작성해야하는지 이해한 것 같다.

Problem : 익숙하지 않은 코드를 마주했을 때 집중력이 급저하되어서 시간을 소요한 부분이 있었던 것 같다.

Try : 최대한 기본기를 탄탄하게 쌓기 위해 더 많은 연습 문제를 풀어봐야 할 것 같다. 그리고 문제나 이후에 분석할 때 최대한 어떤 방향을 원하는지 오해없이 이해하기 위해 문제 및 제안을 어떻게 해결해나갈지 생각하는 시간도 많이 가져야 할 것 같았다.

▼ 아카이브

```
x
```

```
x
```