고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

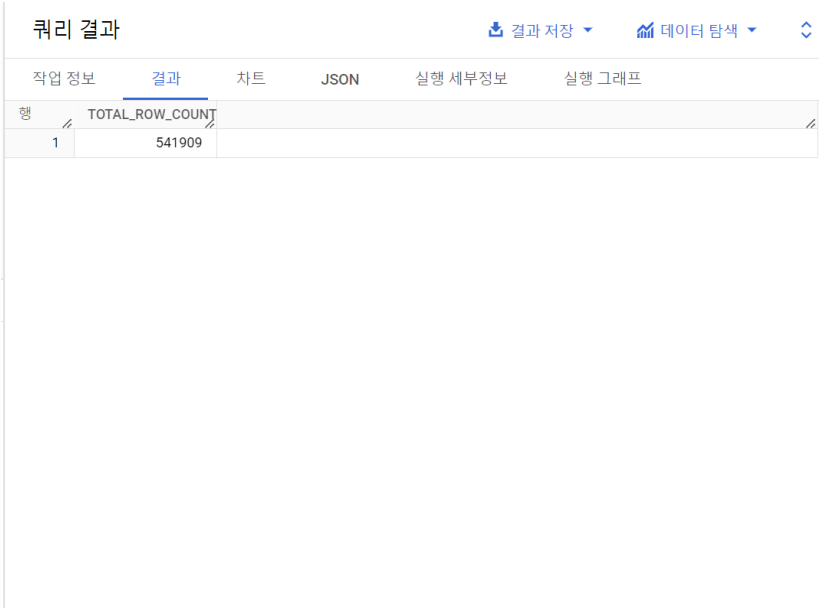
```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.data
LIMIT 10
```

[결과 이미지를 넣어주세요]

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*) AS TOTAL_ROW_COUNT
FROM bright-meridian-439401-g6.modulabs_project1.data
```

[결과 이미지를 넣어주세요]



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 bright-meridian-439401-g6.modulabs_project1.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 column_name, ROUND((total - column_value) / total * 100, 2) AS missing_percentage
FROM
(
    SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM brig
    SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM brig
    SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM
    SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM bright
    SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM
    SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM brig
    SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM br
    SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM bright-m
) AS column_data;
```

[결과 이미지를 넣어주세요]

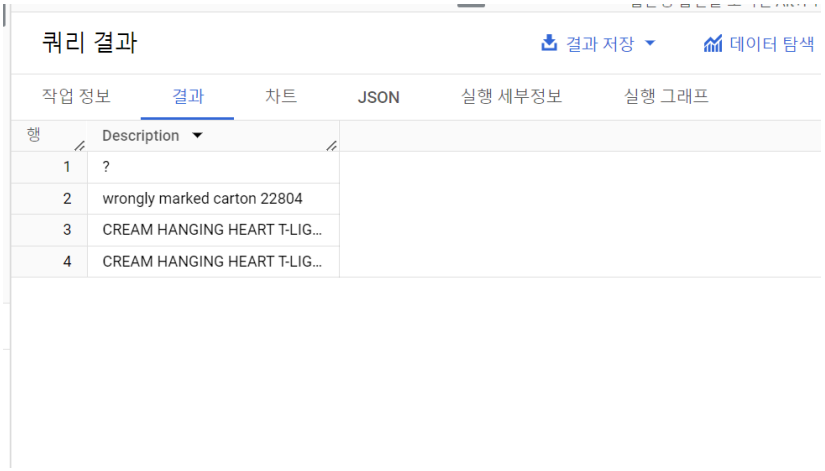
행	column_name	missing_percentage
1	UnitPrice	0.0
2	InvoiceNo	0.0
3	StockCode	0.0
4	Country	0.0
5	InvoiceDate	0.0
6	Quantity	0.0
7	CustomerID	24.93
8	Description	0.27

결측치 처리 전략

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

```
SELECT Description
FROM bright-meridian-439401-g6.modulabs_project1.data
WHERE StockCode = '85123A'
LIMIT 4
```

[결과 이미지를 넣어주세요]



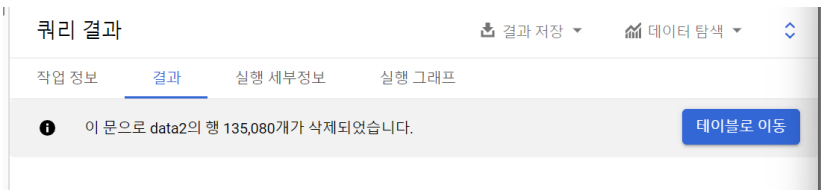
행	Description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	CREAM HANGING HEART T-LIG...

결측치 처리

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

```
DELETE FROM bright-meridian-439401-g6.modulabs_project1.data2
WHERE Description IS NULL
OR CustomerID IS NULL
```

[결과 이미지를 넣어주세요]



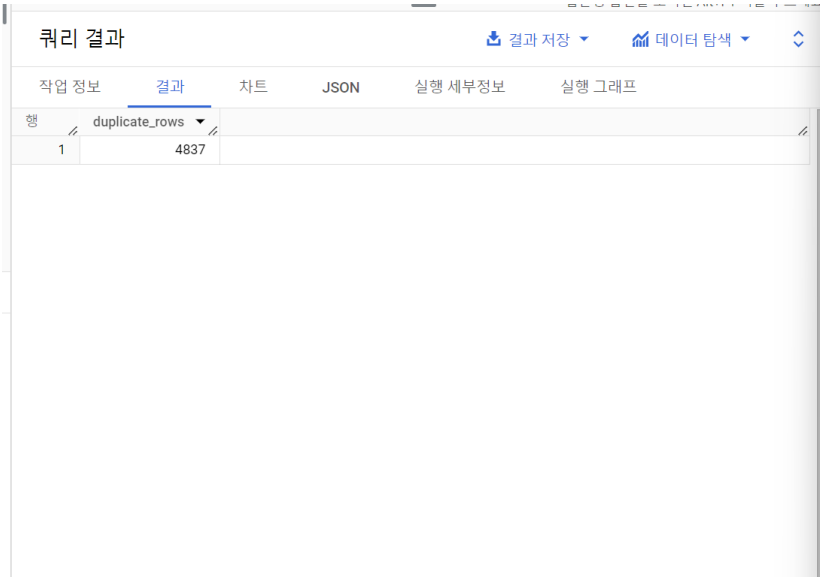
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT COUNT(*) AS duplicate_rows
FROM (
  SELECT InvoiceNo , StockCode , Description , Quantity , InvoiceDate , UnitPrice, CustomerID , Coun
   FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY InvoiceNo , StockCode , Description , Quantity , InvoiceDate , UnitPrice, CustomerID , Coun
HAVING COUNT(*) > 1)
```

[결과 이미지를 넣어주세요]

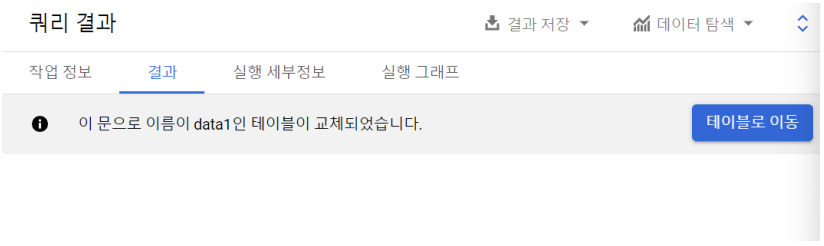


중복값 처리

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

```
CREATE OR REPLACE TABLE `bright-meridian-439401-g6.modulabs_project1.data1` AS(
SELECT DISTINCT *
FROM bright-meridian-439401-g6.modulabs_project1.data1)
```

[결과 이미지를 넣어주세요]



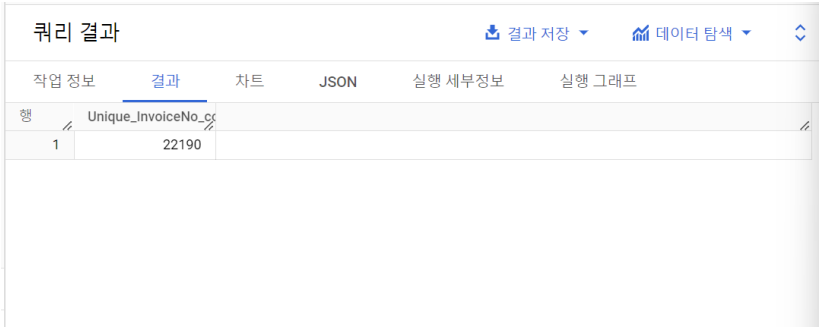
11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

- **고유(unique)한 `InvoiceNo`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS Unique_InvoiceNo_count
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

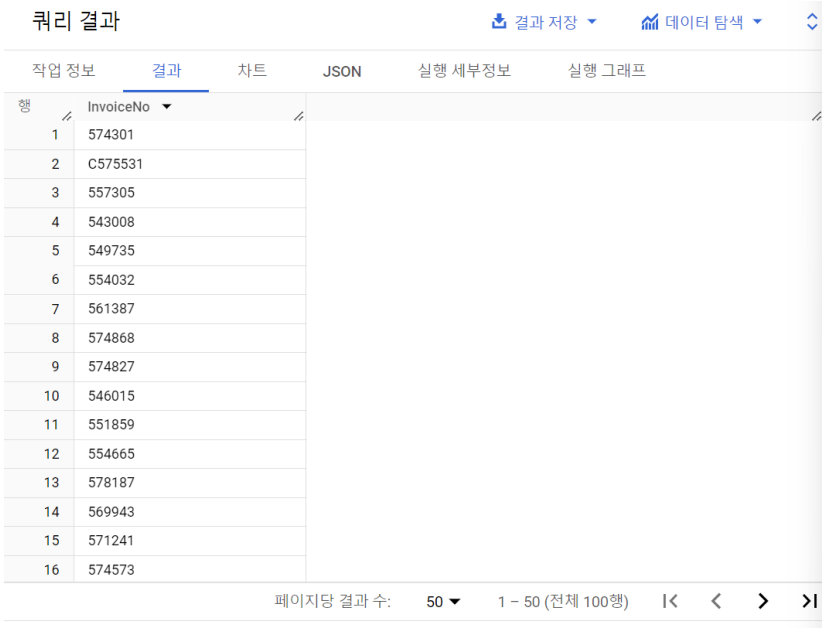
[결과 이미지를 넣어주세요]



- **고유한 `InvoiceNo`를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM bright-meridian-439401-g6.modulabs_project1.data1
LIMIT 100
```

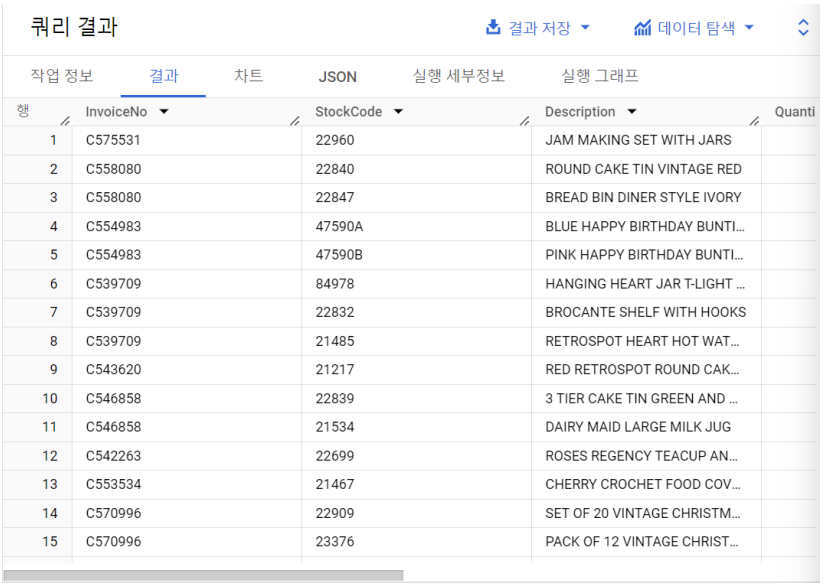
[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]



행	InvoiceNo	StockCode	Description	Quanti
1	C575531	22960	JAM MAKING SET WITH JARS	
2	C558080	22840	ROUND CAKE TIN VINTAGE RED	
3	C558080	22847	BREAD BIN DINER STYLE IVORY	
4	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	
5	C554983	47590B	PINK HAPPY BIRTHDAY BUNTI...	
6	C539709	84978	HANGING HEART JAR T-LIGHT ...	
7	C539709	22832	BROCANTE SHELF WITH HOOKS	
8	C539709	21485	RETROSPOT HEART HOT WAT...	
9	C543620	21217	RED RETROSPOT ROUND CAK...	
10	C546858	22839	3 TIER CAKE TIN GREEN AND ...	
11	C546858	21534	DAIRY MAID LARGE MILK JUG	
12	C542263	22699	ROSES REGENCY TEACUP AN...	
13	C553534	21467	CHERRY CROCHET FOOD COV...	
14	C570996	22909	SET OF 20 VINTAGE CHRISTM...	
15	C570996	23376	PACK OF 12 VINTAGE CHRIST...	

- **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 Can
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

행	Cancled_Rate
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

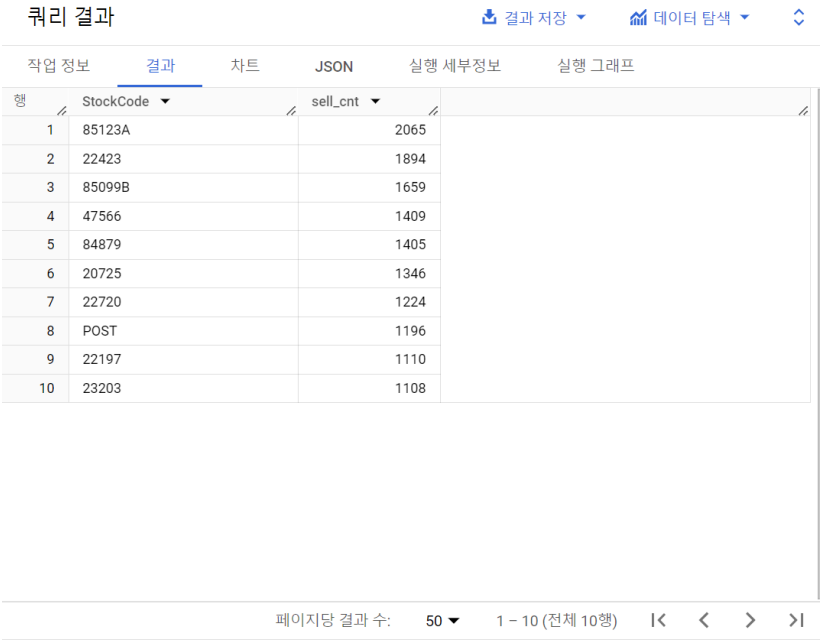
```
SELECT COUNT(DISTINCT stockcode) AS stockcode_count
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY stockcode
ORDER BY sell_cnt DESC
LIMIT 10
```

[결과 이미지를 넣어주세요]

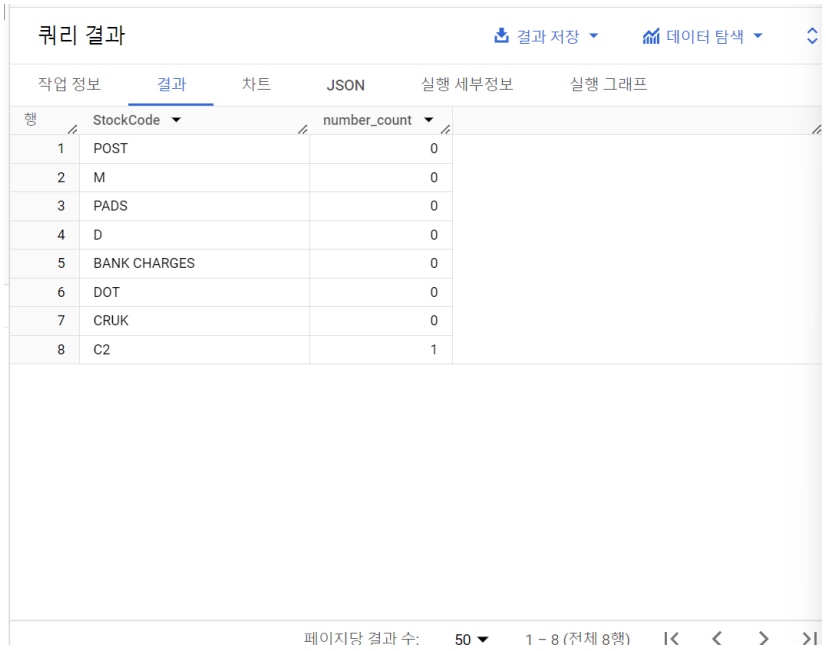


- **`StockCode` 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**

- **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 bright-meridian-439401-g6.modulabs_project1.data1
)
WHERE number_count < 2
```

[결과 이미지를 넣어주세요]



- `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 ROUND(SUM(CASE WHEN number_count < 2 THEN 1 ELSE 0 END)/ COUNT(number_count) *100,  2) AS num
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM bright-meridian-439401-g6.modulabs_project1.data1
)
```

[결과 이미지를 넣어주세요]

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM bright-meridian-439401-g6.modulabs_project1.data1 )
  WHERE number_count < 2
);
```

[결과 이미지를 넣어주세요]

`Description` 살펴보기

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

```
SELECT Description , COUNT(*) AS description_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY Description
LIMIT 30
```

[결과 이미지를 넣어주세요]

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE Description LIKE '%Next Day Carriage%'
AND  Description LIKE "%High Resolution Image%"
```

[결과 이미지를 넣어주세요]

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.data1 AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

`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 bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

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

```
SELECT COUNTIF(quantity = 0) AS cnt_quantity,MIN(quantity) AS min_quantity, MAX(quantity) AS max_qua
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.data1  AS
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE Unitprice != 0
```

[결과 이미지를 넣어주세요]

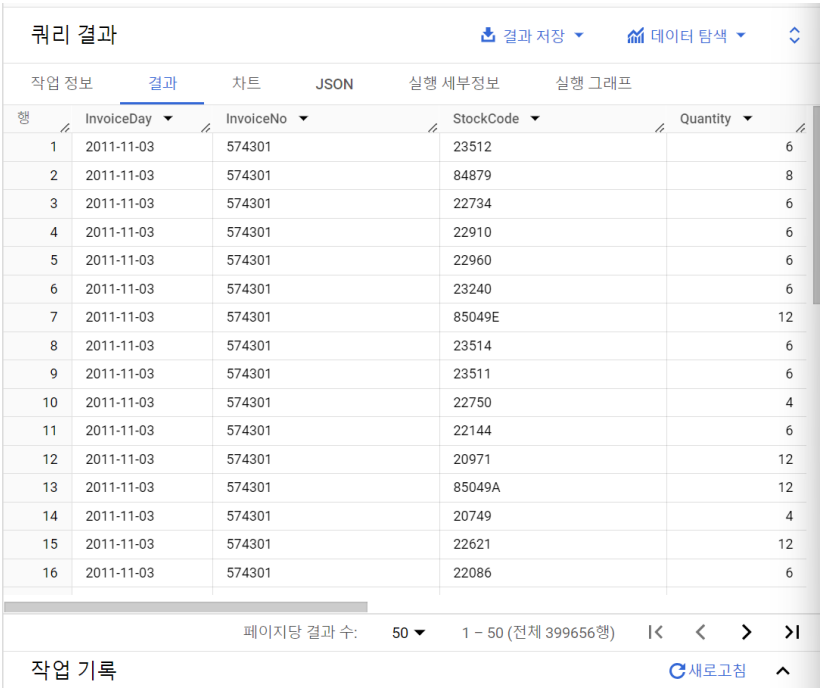
11-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

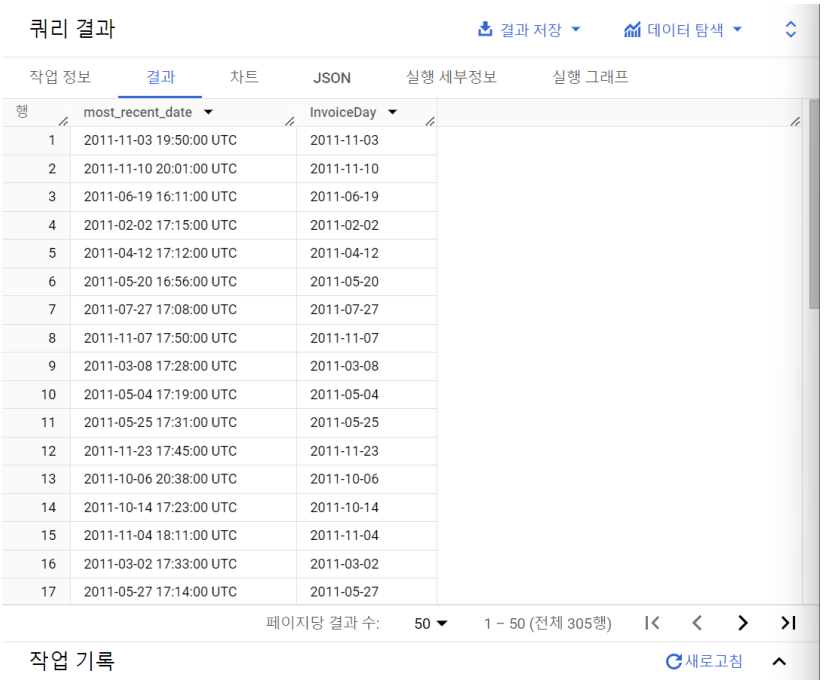


행	InvoiceDay	InvoiceNo	StockCode	Quantity
1	2011-11-03	574301	23512	6
2	2011-11-03	574301	84879	8
3	2011-11-03	574301	22734	6
4	2011-11-03	574301	22910	6
5	2011-11-03	574301	22960	6
6	2011-11-03	574301	23240	6
7	2011-11-03	574301	85049E	12
8	2011-11-03	574301	23514	6
9	2011-11-03	574301	23511	6
10	2011-11-03	574301	22750	4
11	2011-11-03	574301	22144	6
12	2011-11-03	574301	20971	12
13	2011-11-03	574301	85049A	12
14	2011-11-03	574301	20749	4
15	2011-11-03	574301	22621	12
16	2011-11-03	574301	22086	6

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
   MAX(InvoiceDate) AS most_recent_date,
   DATE(InvoiceDate) AS InvoiceDay
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY DATE(InvoiceDate);
```

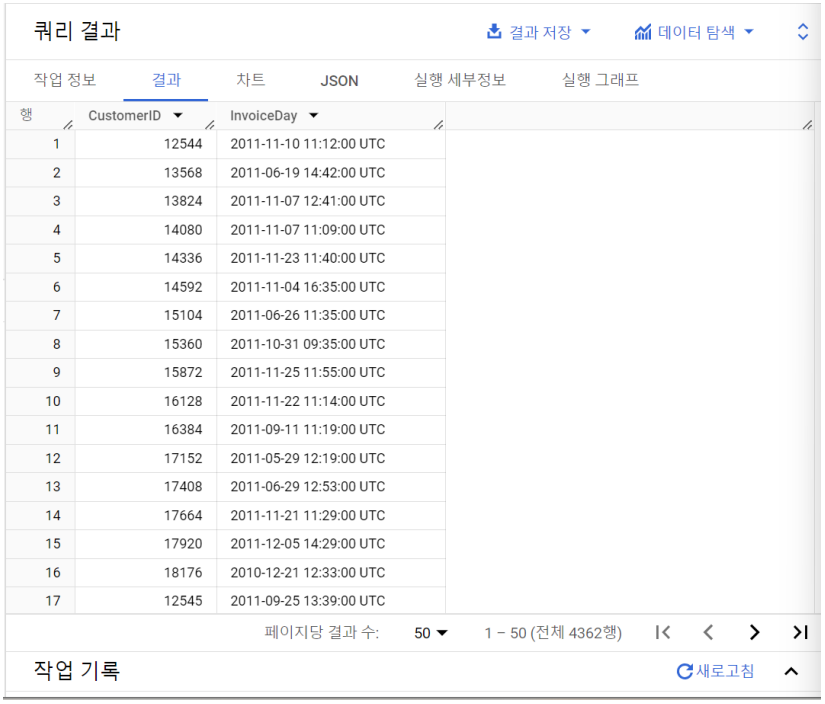
[결과 이미지를 넣어주세요]



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

```
SELECT
    CustomerID,
    MAX(InvoiceDate) AS InvoiceDay
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY  CustomerID
```

[결과 이미지를 넣어주세요]

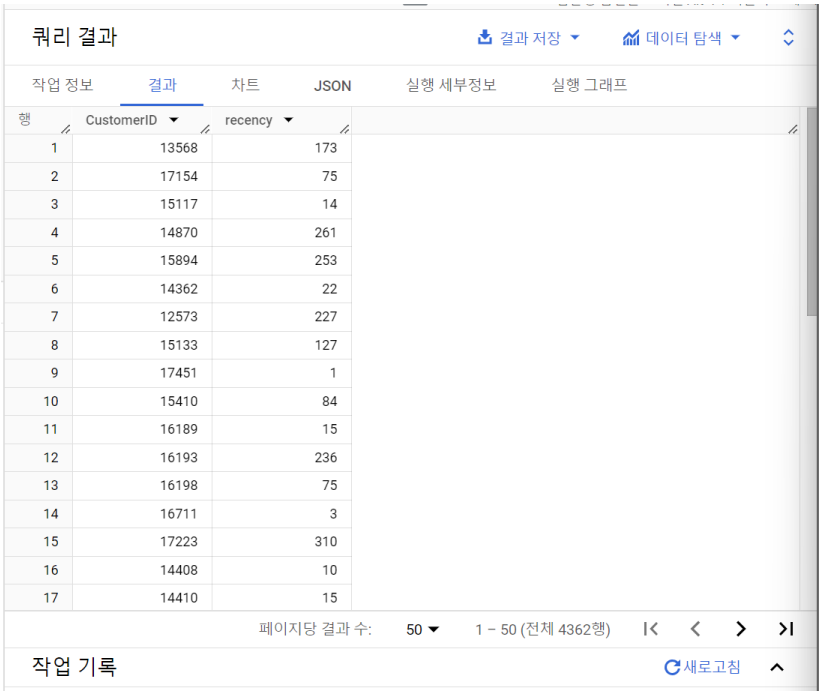


- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

```
SELECT
  CustomerID,
```

```
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM bright-meridian-439401-g6.modulabs_project1.data1
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.user_r AS
 (SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM bright-meridian-439401-g6.modulabs_project1.data1
  GROUP BY CustomerID
))
```

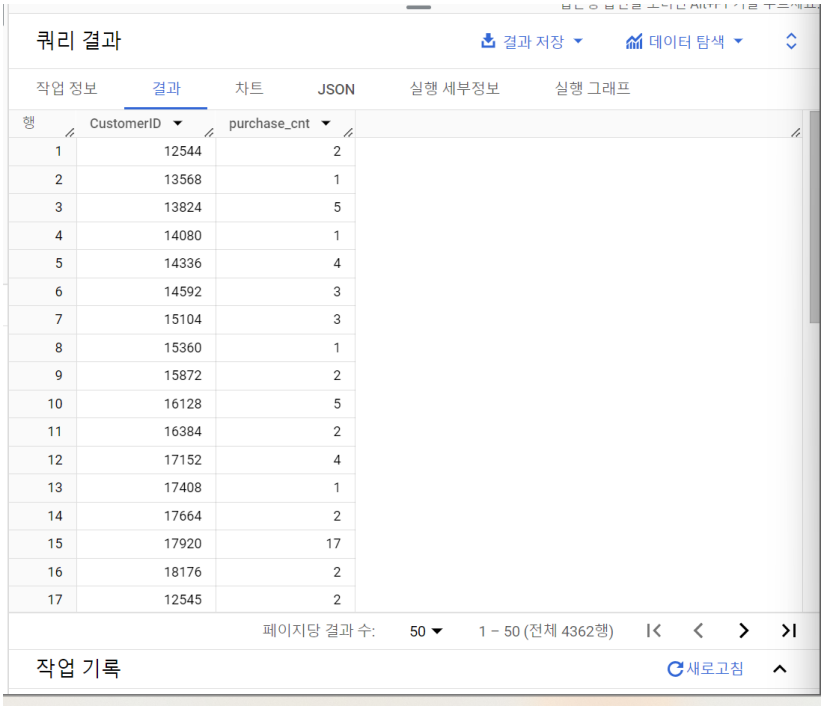
[결과 이미지를 넣어주세요]

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]



- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
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 bright-meridian-439401-g6.modulabs_project1.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

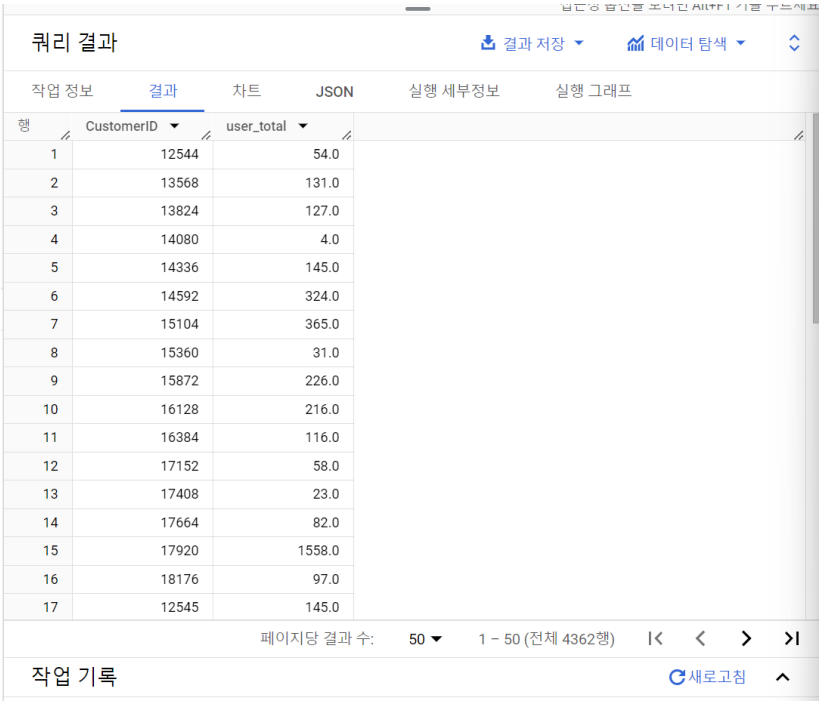
[결과 이미지를 넣어주세요]

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice),0) AS user_total
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]



- **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 bright-meridian-439401-g6.modulabs_project1.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(user_total / purchase_cnt,0) AS user_average
FROM bright-meridian-439401-g6.modulabs_project1.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice),0) AS user_total
    FROM bright-meridian-439401-g6.modulabs_project1.data1
    GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

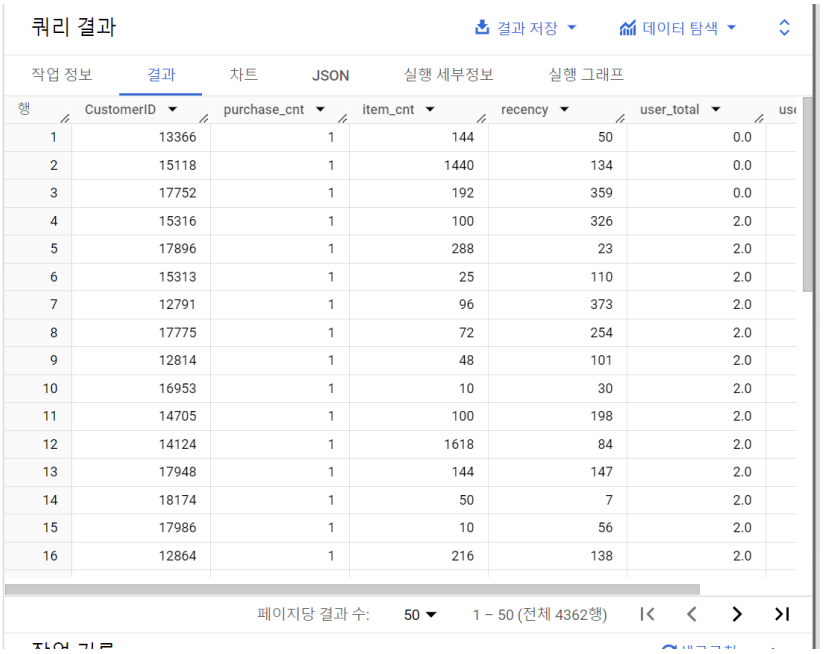
[결과 이미지를 넣어주세요]

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.user_rfm
```

[결과 이미지를 넣어주세요]



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 bright-meridian-439401-g6.modulabs_project1.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
      bright-meridian-439401-g6.modulabs_project1.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM bright-meridian-439401-g6.modulabs_project1.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

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 bright-meridian-439401-g6.modulabs_project1.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    SUM(CASE WHEN stockcode LIKE 'C%'THEN 1 ELSE 0 END) AS cancel_frequency
  FROM bright-meridian-439401-g6.modulabs_project1.data
  GROUP BY customerID
)
```

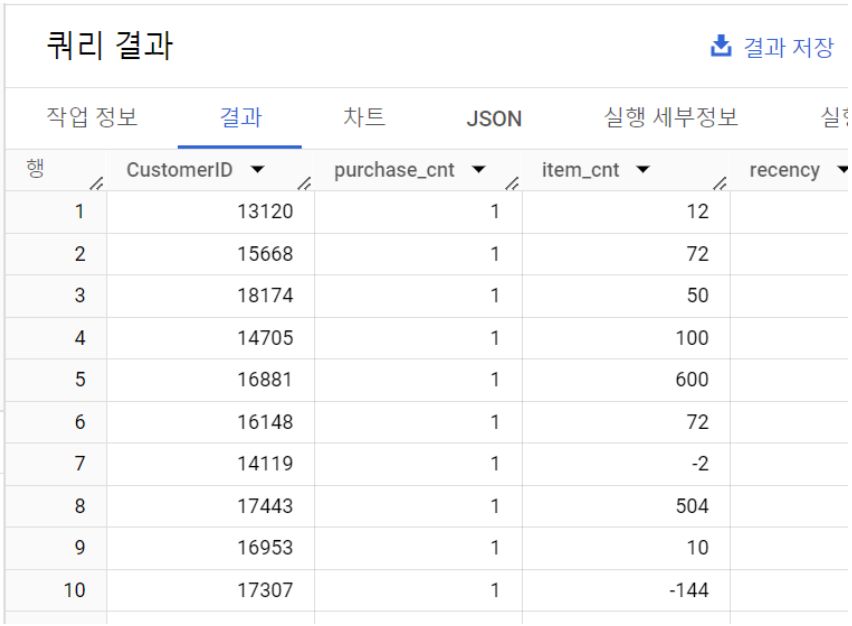
```
SELECT u.*, t.* EXCEPT(CustomerID),
      ROUND(cancel_frequency * 100.0 / total_transactions, 2)
      AS cancel_rate
FROM bright-meridian-439401-g6.modulabs_project1.user_data AS u
LEFT JOIN TransactionInfo AS t
ON  U.customerID = t.customerID
```

[결과 이미지를 넣어주세요]

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

```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.user_data
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency
1	13120	1	12	
2	15668	1	72	
3	18174	1	50	
4	14705	1	100	
5	16881	1	600	
6	16148	1	72	
7	14119	1	-2	
8	17443	1	504	
9	16953	1	10	
10	17307	1	-144	

회고

[회고 내용을 작성해주세요]

처음에 SQL 기초 문법을 배울때는 파이썬보다 쉽다 생각했었는데 이번 퀘스트를 하면서
얼마나 숙지해야 할 내용들이 많고 복습이 꼭 필요하다는 것을 깨달았습니다.
GPT나 배웠던 내용들을 보면서 여차여차 끝내긴 했는데 정말 공부를 많이 해야하고,
쉬운 부분이 있다고 거만하지 말고, 항상 겸손한 자세가 필요하다는 것을 깨달았습니다....