고객을 세그먼테이션하자 [프로젝트]

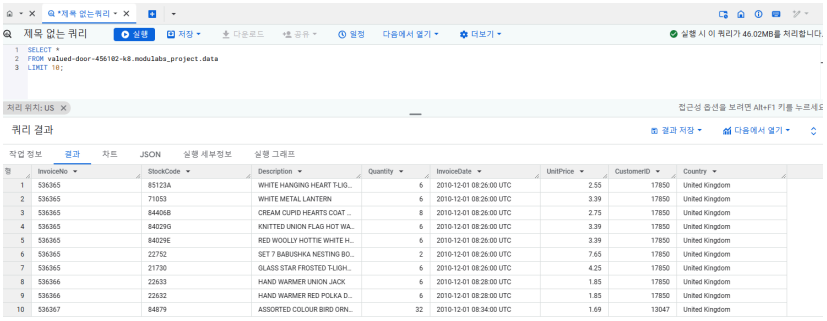
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM valued-door-456102-k8.modulabs_project.data
LIMIT 10;
```

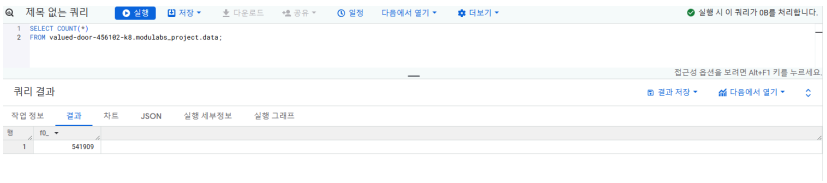
[결과 이미지를 넣어주세요]



- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM valued-door-456102-k8.modulabs_project.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 valued-door-456102-k8.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

11-4. 데이터 전처리 방법(1): 결측치 제거

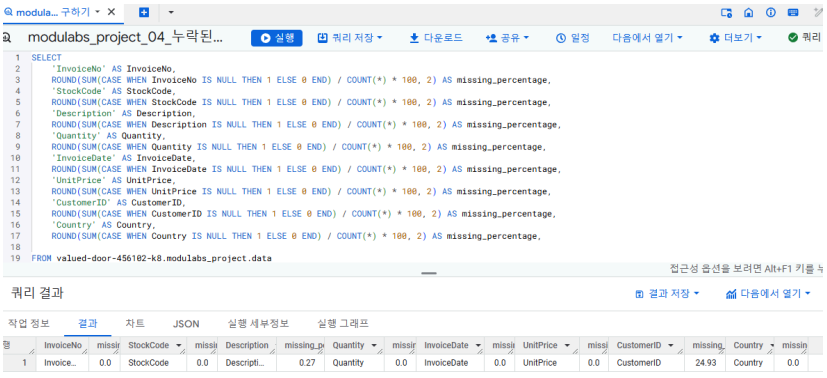
컬럼 별 누락된 값의 비율 계산

- **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,
  'StockCode' AS StockCode,
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage,
  'Description' AS Description,
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage,
  'Quantity' AS Quantity,
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage,
  'InvoiceDate' AS InvoiceDate,
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage,
  'UnitPrice' AS UnitPrice,
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage,
  'CustomerID' AS CustomerID,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage,
  'Country' AS Country,
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage,

FROM valued-door-456102-k8.modulabs_project.data
```

[결과 이미지를 넣어주세요]



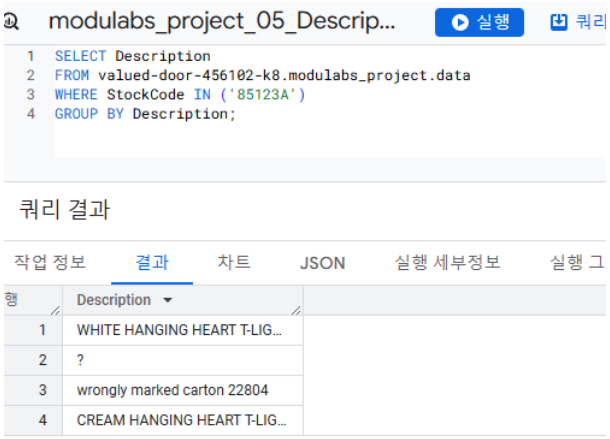
결측치 처리 전략

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

```
SELECT Description
FROM valued-door-456102-k8.modulabs_project.data
```

```
WHERE StockCode IN ('85123A')
GROUP BY Description;
```

[결과 이미지를 넣어주세요]

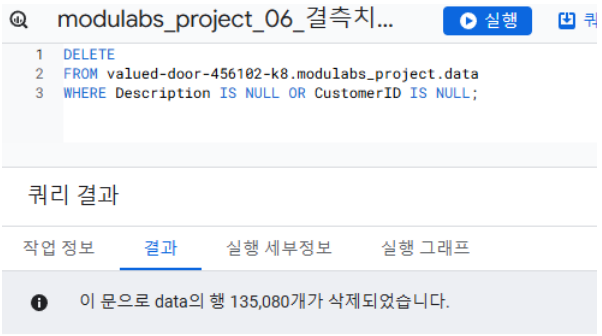


결측치 처리

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

```
DELETE
FROM valued-door-456102-k8.modulabs_project.data
WHERE Description IS NULL OR CustomerID IS NULL;
```

[결과 이미지를 넣어주세요]



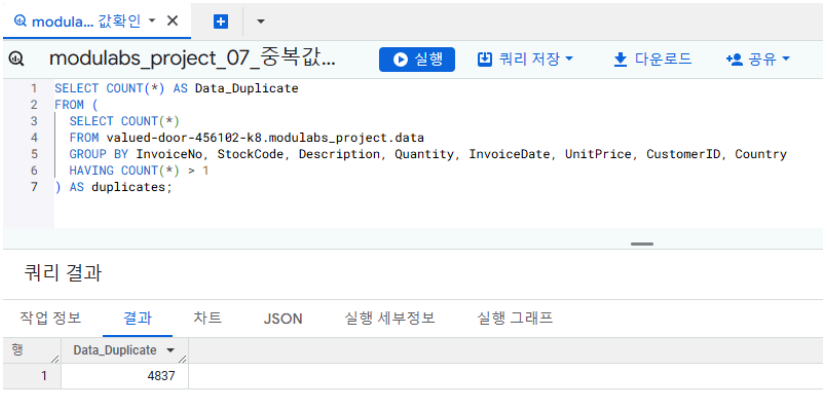
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT COUNT(*) AS Data_Duplicate
FROM (
  SELECT COUNT(*)
  FROM valued-door-456102-k8.modulabs_project.data
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
) AS duplicates;
```

[결과 이미지를 넣어주세요]

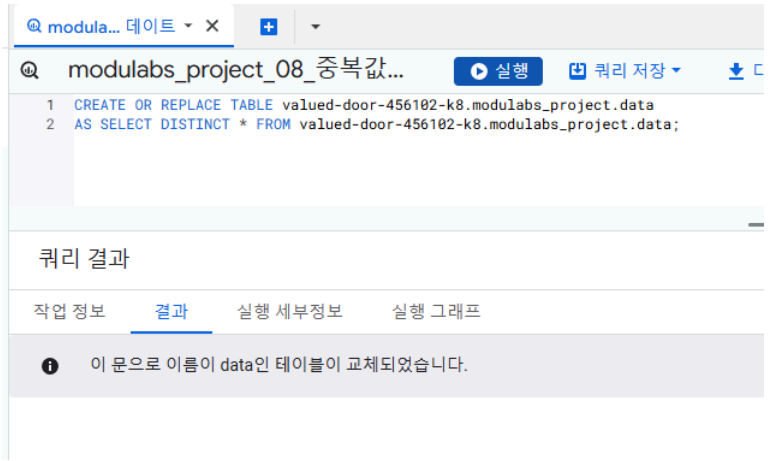


중복값 처리

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

```
CREATE OR REPLACE TABLE valued-door-456102-k8.modulabs_project.data
AS SELECT DISTINCT * FROM valued-door-456102-k8.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



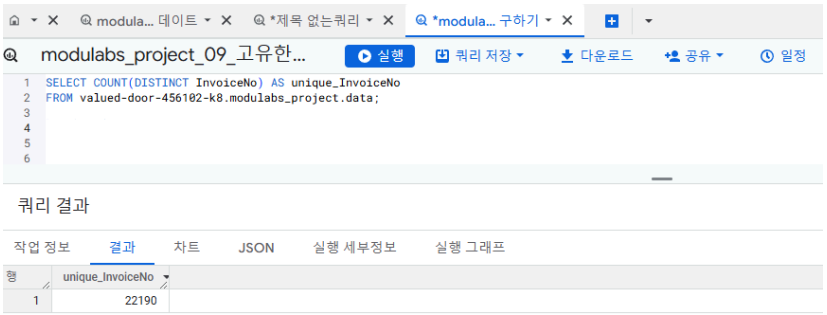
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_InvoiceNo
FROM valued-door-456102-k8.modulabs_project.data;
```

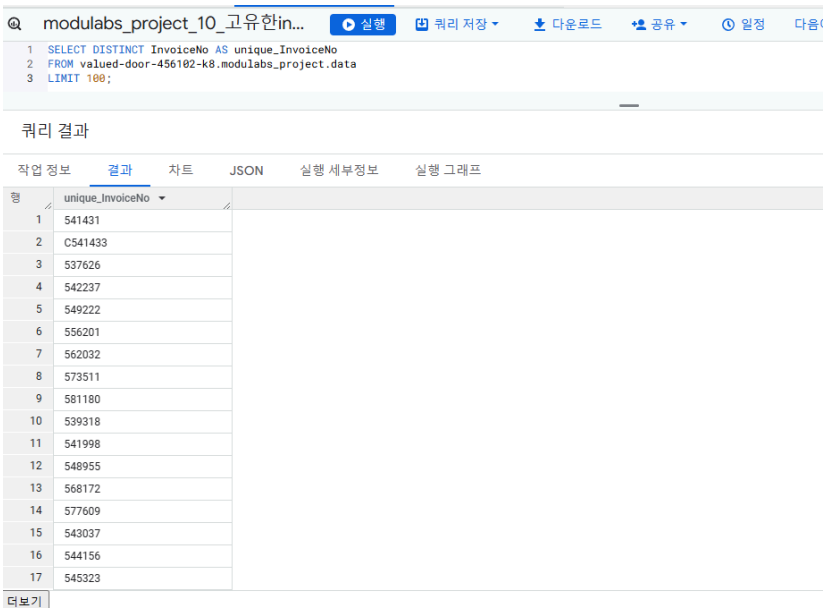
[결과 이미지를 넣어주세요]



- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo AS unique_InvoiceNo
FROM valued-door-456102-k8.modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

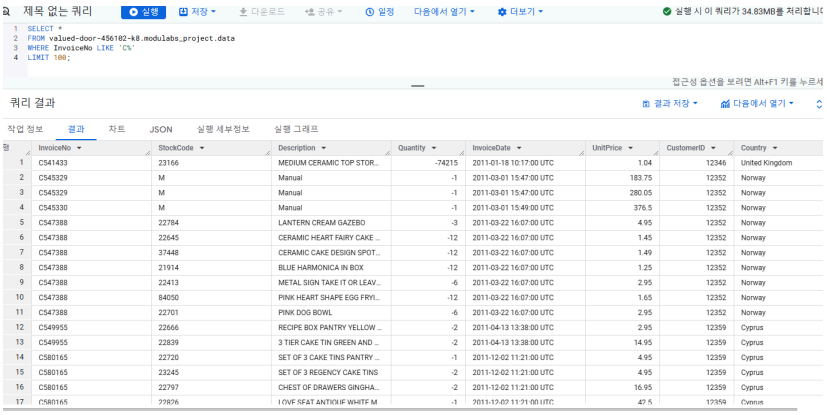


- **InvoiceNo가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)**

```
SELECT *
FROM valued-door-456102-k8.modulabs_project.data
```

```
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

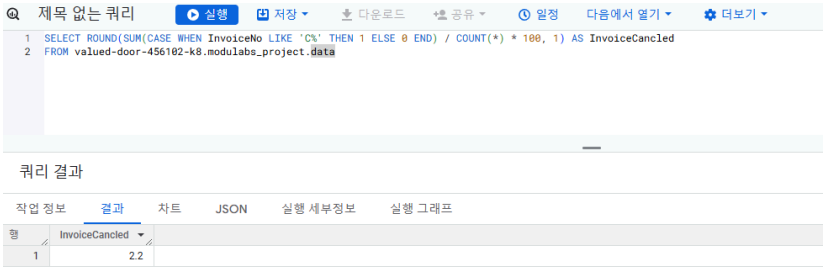
[결과 이미지를 넣어주세요]



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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*) * 100, 1) AS InvoiceCancled
FROM valued-door-456102-k8.modulabs_project.data
```

[결과 이미지를 넣어주세요]

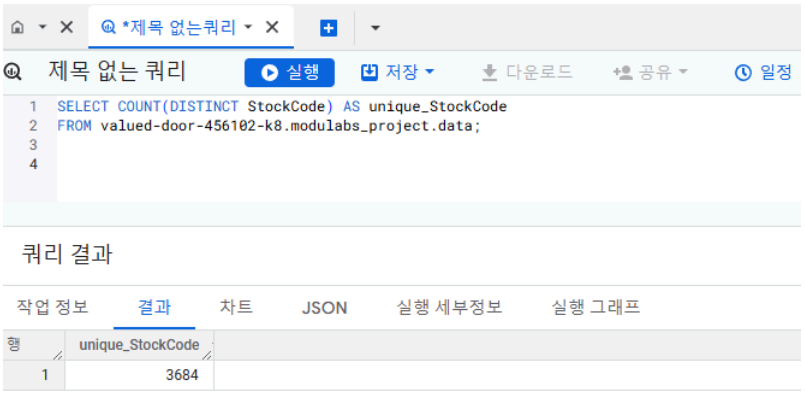


`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_StockCode
FROM valued-door-456102-k8.modulabs_project.data;
```

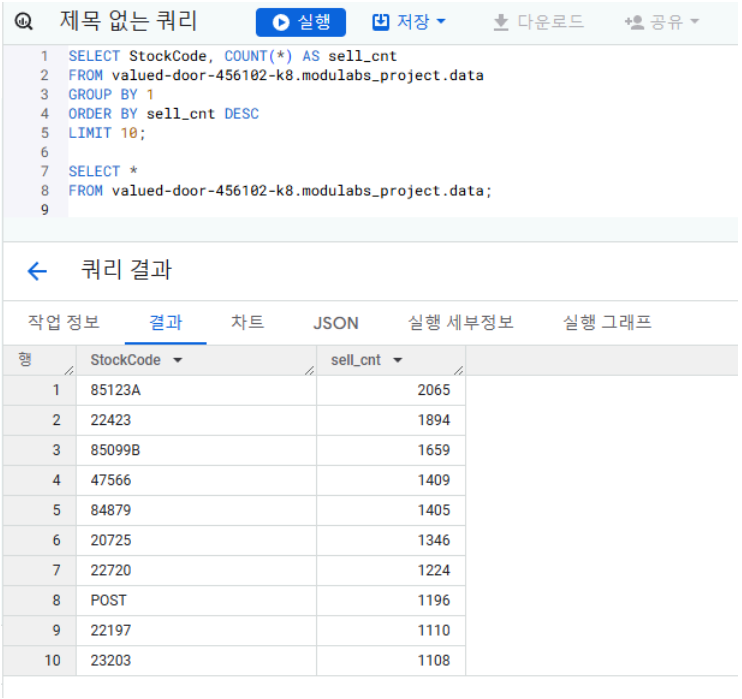
[결과 이미지를 넣어주세요]



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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM valued-door-456102-k8.modulabs_project.data
GROUP BY 1
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

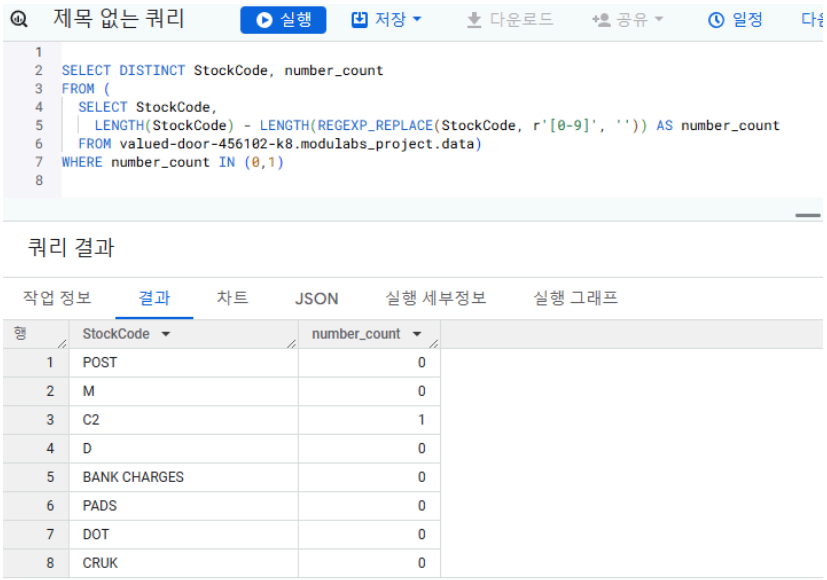


행	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
9	22197	1110
10	23203	1108

- **`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 valued-door-456102-k8.modulabs_project.data)
WHERE number_count IN (0,1)
```

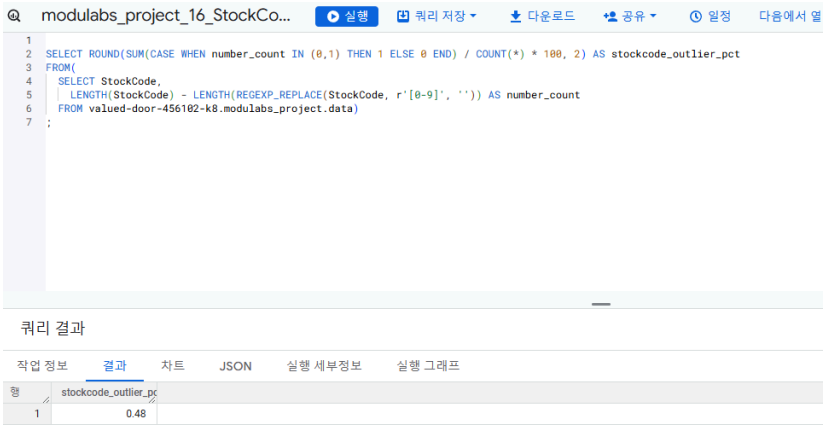
[결과 이미지를 넣어주세요]



- `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 IN (0,1) THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS stockcode_outlier_pct
FROM(
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM valued-door-456102-k8.modulabs_project.data);
```

[결과 이미지를 넣어주세요]

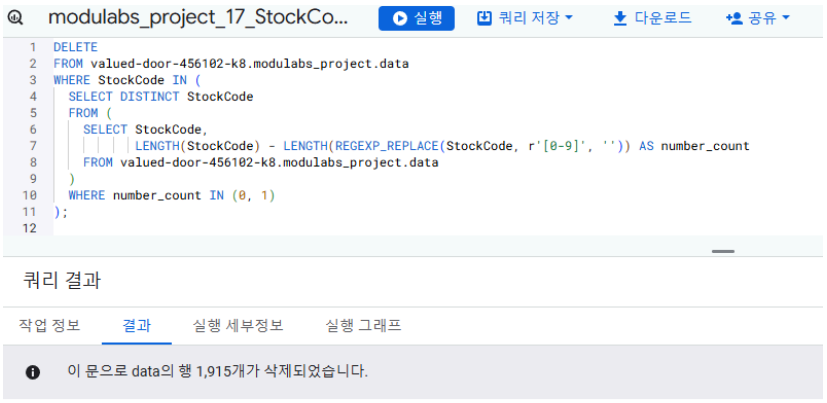


- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE
FROM valued-door-456102-k8.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 valued-door-456102-k8.modulabs_project.data
  )
  WHERE number_count IN (0, 1)
);
```

[결과 이미지를 넣어주세요]



Description 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM valued-door-456102-k8.modulabs_project.data
GROUP BY Description
LIMIT 30;
```

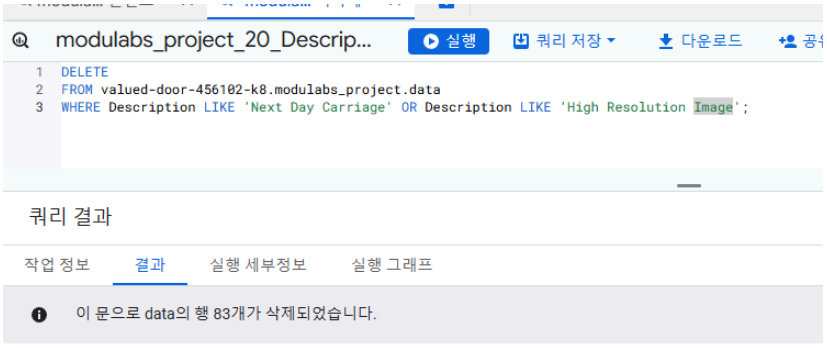
[결과 이미지를 넣어주세요]

	Description	description_cnt
1	MEDIUM CERAMIC TOP STORAGE JAR	208
2	RED TOADSTOOL LED NIGHT LIGHT	539
3	SET/3 DECOUPAGE STACKING TINS	54
4	FOUR HOOK WHITE LOVEBIRDS	265
5	ALARM CLOCK BAKELIKE CHOCOLATE	339
6	BLACK EAR MUFF HEADPHONES	18
7	BLUE 3 PIECE POLKADOT CUTLERY SET	97
8	EMERGENCY FIRST AID TIN	126
9	COLOUR GLASS. STAR T-LIGHT HOLDER	247
10	RED 3 PIECE RETROSPOT CUTLERY SET	100
11	SET OF 2 TINS VINTAGE BATHROOM	59
12	LARGE HEART MEASURING SPOONS	226
13	RED DRAWER KNOB ACRYLIC EDWARDIAN	101
14	BATHROOM METAL SIGN	60
15	CLEAR DRAWER KNOB ACRYLIC EDWARDIAN	331
16	BOOM BOX SPEAKER BOYS	56
17	CAMOUFLAGE EAR MUFF HEADPHONES	17

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM valued-door-456102-k8.modulabs_project.data
WHERE Description LIKE 'Next Day Carriage' OR Description LIKE 'High Resolution Image';
```

[결과 이미지를 넣어주세요]

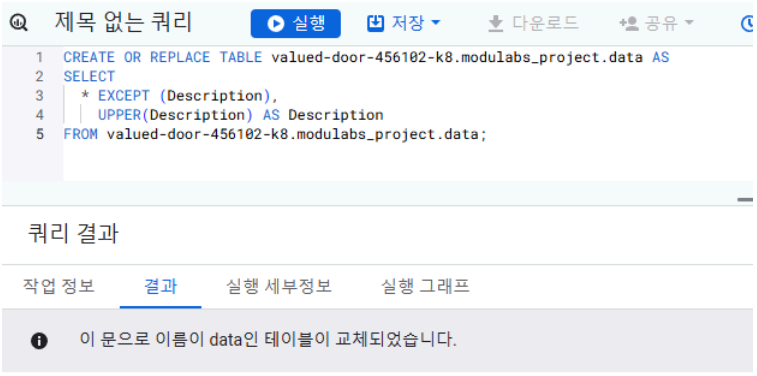


- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE valued-door-456102-k8.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
```

```
    UPPER(Description) AS Description
FROM valued-door-456102-k8.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

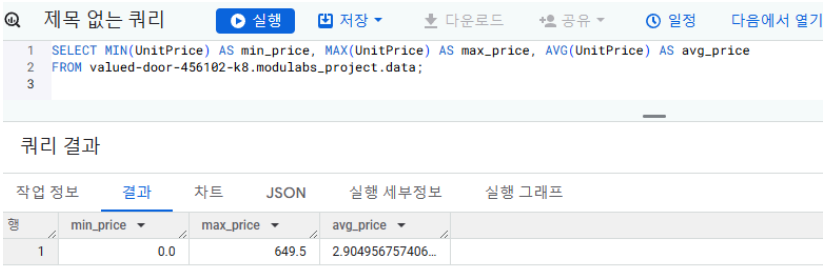


UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM valued-door-456102-k8.modulabs_project.data;
```

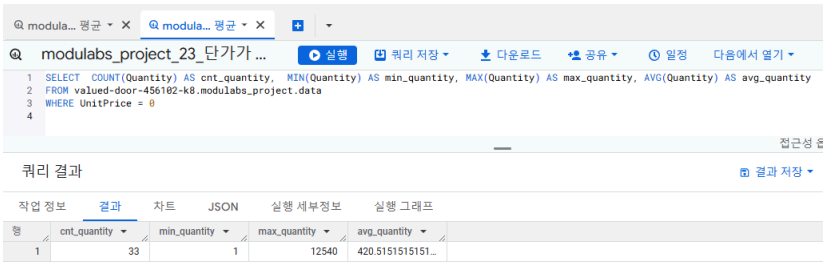
[결과 이미지를 넣어주세요]



- **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
FROM valued-door-456102-k8.modulabs_project.data
WHERE UnitPrice = 0
```

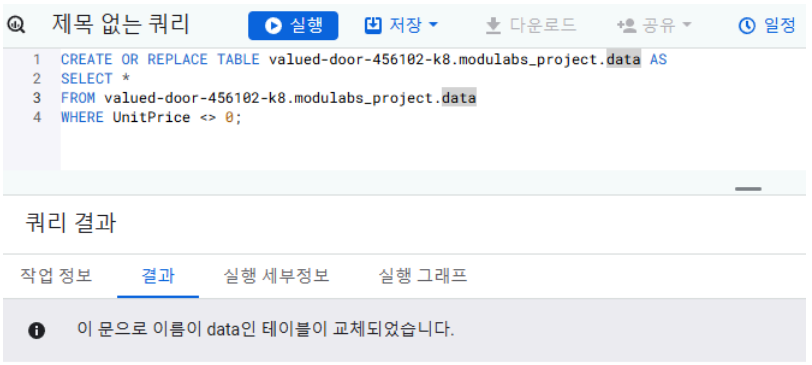
[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE valued-door-456102-k8.modulabs_project.data AS
SELECT *
FROM valued-door-456102-k8.modulabs_project.data
WHERE UnitPrice <> 0;
```

[결과 이미지를 넣어주세요]



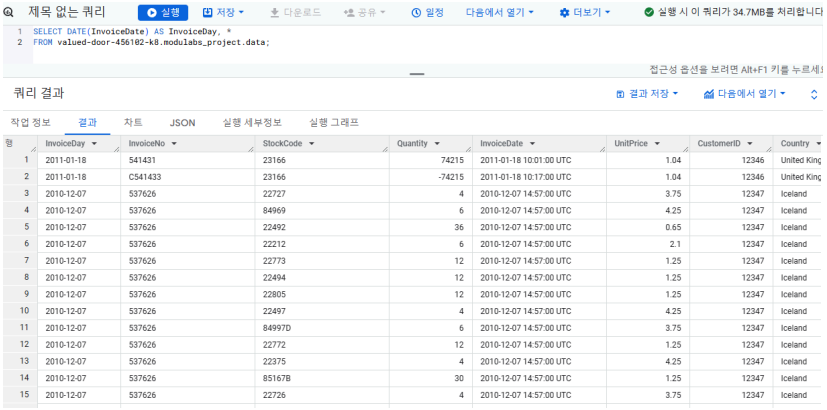
11-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM valued-door-456102-k8.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

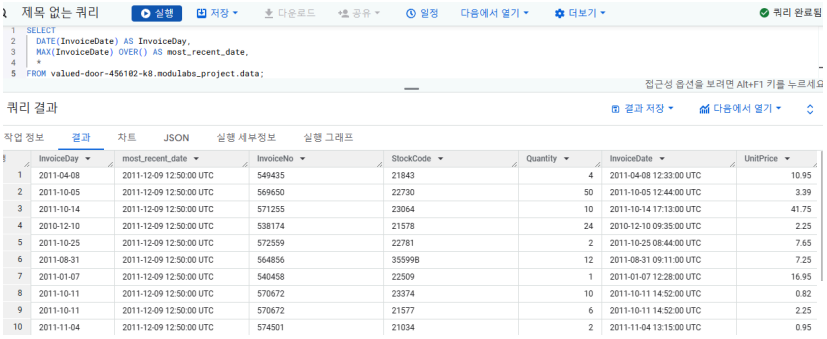


행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United King
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United King
3	2010-12-07	537626	22727	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
4	2010-12-07	537626	84969	6	2010-12-07 14:57:00 UTC	4.25	12347	Iceland
5	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland
6	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC	2.1	12347	Iceland
7	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
8	2010-12-07	537626	22494	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
9	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
10	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland
11	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
12	2010-12-07	537626	22772	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
13	2010-12-07	537626	22375	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland
14	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
15	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  DATE(InvoiceDate) AS InvoiceDay,
  MAX(InvoiceDate) OVER() AS most_recent_date
FROM valued-door-456102-k8.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

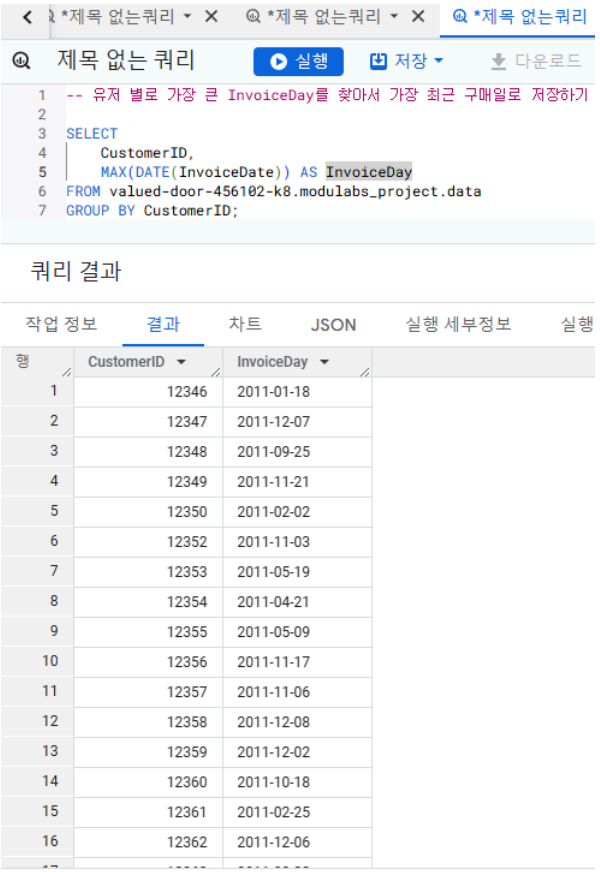


	InvoiceDay	most_recent_date	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice
1	2011-04-08	2011-12-09 12:50:00 UTC	549435	21843	4	2011-04-08 12:33:00 UTC	10.95
2	2011-10-05	2011-12-09 12:50:00 UTC	569650	22730	50	2011-10-05 12:44:00 UTC	3.39
3	2011-10-14	2011-12-09 12:50:00 UTC	571255	23064	10	2011-10-14 17:13:00 UTC	41.75
4	2010-12-10	2011-12-09 12:50:00 UTC	538174	21578	24	2010-12-10 09:35:00 UTC	2.25
5	2011-10-25	2011-12-09 12:50:00 UTC	572559	22781	2	2011-10-25 08:44:00 UTC	7.65
6	2011-08-31	2011-12-09 12:50:00 UTC	564856	35599B	12	2011-08-31 09:11:00 UTC	7.25
7	2011-01-07	2011-12-09 12:50:00 UTC	540458	22509	1	2011-01-07 12:28:00 UTC	16.95
8	2011-10-11	2011-12-09 12:50:00 UTC	570672	23374	10	2011-10-11 14:52:00 UTC	0.82
9	2011-10-11	2011-12-09 12:50:00 UTC	570672	21577	6	2011-10-11 14:52:00 UTC	2.25
10	2011-11-04	2011-12-09 12:50:00 UTC	574501	21034	2	2011-11-04 13:15:00 UTC	0.95

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

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM valued-door-456102-k8.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]

제목 없는 쿼리 실행 저장 다운로드 공유

```
1
2  --가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산
3
4  SELECT
5    CustomerID,
6    EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
7  FROM (
8    SELECT
9      CustomerID,
10     MAX(DATE(InvoiceDate)) AS InvoiceDay
11   FROM valued-door-456102-k8.modulabs_project.data
12   GROUP BY CustomerID
```

쿼리 결과

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

행	CustomerID	recency
1	12432	42
2	12506	232
3	12895	42
4	12965	89
5	13269	1
6	13271	37
7	13493	275
8	13518	85
9	13658	9
10	13824	32
11	13859	326
12	13901	72
13	14092	7

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

```
CREATE OR REPLACE TABLE valued-door-456102-k8.modulabs_project.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM valued-door-456102-k8.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
CustomerID,
COUNT(InvoiceNo) AS purchase_cnt
FROM valued-door-456102-k8.modulabs_project.data
GROUP BY CustomerID;
```

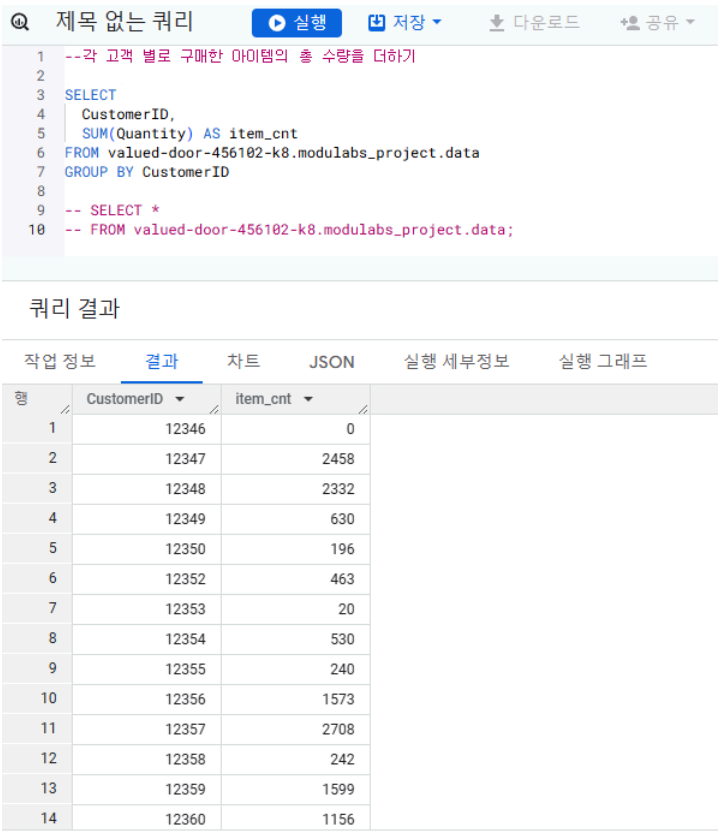
[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84
7	12353	4
8	12354	58
9	12355	13
10	12356	58
11	12357	131
12	12358	17
13	12359	251
14	12360	126
15	12361	9
16	12362	264

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM valued-door-456102-k8.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]



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

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

```
CREATE OR REPLACE TABLE valued-door-456102-k8.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM valued-door-456102-k8.modulabs_project.data
GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM valued-door-456102-k8.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 valued-door-456102-k8.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

```
CREATE OR REPLACE TABLE valued-door-456102-k8.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM valued-door-456102-k8.modulabs_project.data
GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM valued-door-456102-k8.modulabs_project.data
GROUP BY CustomerID
)
```

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

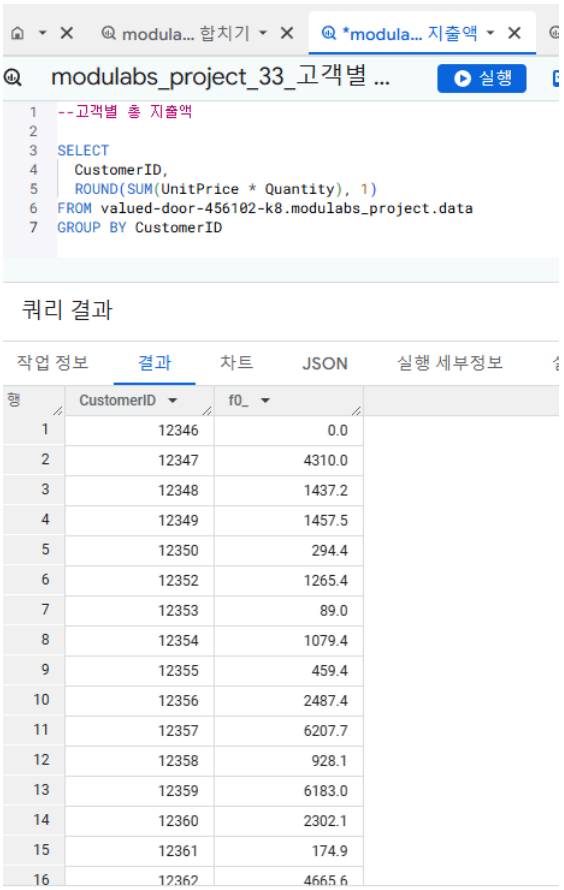
이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1)
FROM valued-door-456102-k8.modulabs_project.data
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 valued-door-456102-k8.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, 1) AS user_average
FROM valued-door-456102-k8.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
  FROM valued-door-456102-k8.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

제목 없는 쿼리 실행 저장 다운로드 공유

```sql
CREATE OR REPLACE TABLE valued-door-456102-k8.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, 1) AS user_average
FROM valued-door-456102-k8.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
  FROM valued-door-456102-k8.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```sql
SELECT *
FROM valued-door-456102-k8.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

modulabs_project_35_최종 use... 실행 쿼리 저장 다운로드 공유 일정 다음에서 열기 더보기

```sql
SELECT *
FROM valued-door-456102-k8.modulabs_project.user_rfm
```

쿼리 결과 결과 저장

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	18102	431	64124	0	259657.3	602.5
2	17428	343	9435	0	17078.5	49.8
3	16705	284	5458	0	13946.1	49.1
4	16558	474	5346	0	8257.0	17.4
5	12662	222	2023	0	3511.1	15.8
6	14051	214	3740	0	15462.3	72.3
7	17389	223	7442	0	31317.5	140.4
8	12748	4440	23516	0	29820.0	6.7
9	14441	59	430	0	1545.1	26.2
10	12433	420	11071	0	13375.9	31.8
11	15694	78	1664	0	6408.6	82.2
12	17364	409	2671	0	4437.2	10.8
13	12526	68	624	0	1172.7	17.2
14	17315	482	3805	0	6153.3	12.8
15	17001	169	2164	0	3989.6	23.6
16	13426	158	2225	0	3558.3	22.5
17	17581	451	5849	0	10716.3	23.8
18	15804	273	2513	0	3848.5	14.1

페이지당 결과 수: 50 1 – 50 (전체 4362행)

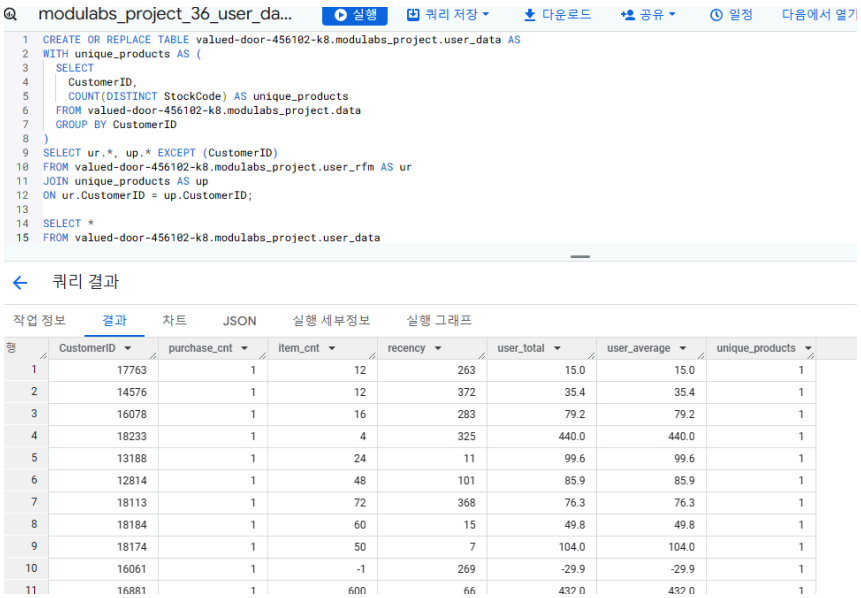
11-8. 추가 Feature 추출

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]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	17763	1	12	263	15.0	15.0	1
2	14576	1	12	372	35.4	35.4	1
3	16078	1	16	283	79.2	79.2	1
4	18233	1	4	325	440.0	440.0	1
5	13188	1	24	11	99.6	99.6	1
6	12814	1	48	101	85.9	85.9	1
7	18113	1	72	368	76.3	76.3	1
8	18184	1	60	15	49.8	49.8	1
9	18174	1	50	7	104.0	104.0	1
10	16061	1	-1	269	-29.9	-29.9	1
11	16881	1	600	66	432.0	432.0	1

2. 평균 구매 주기

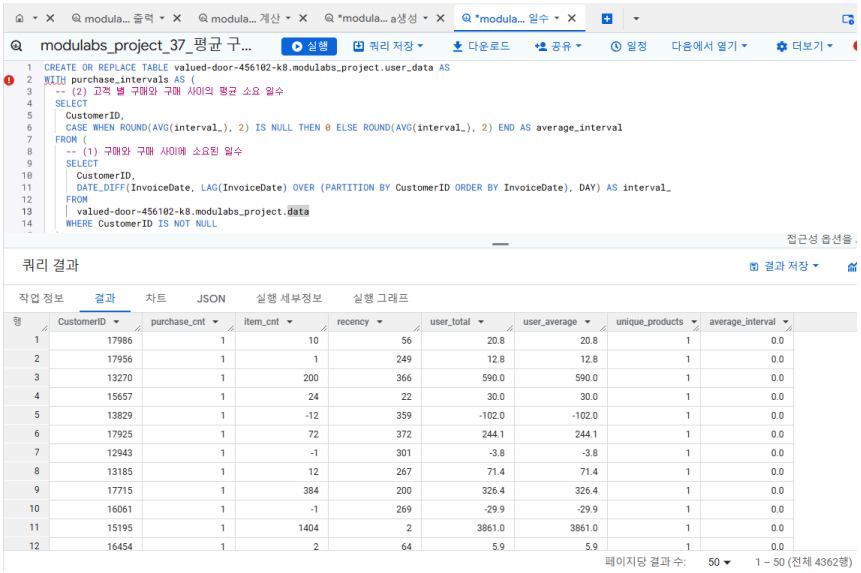
- **고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)**
  - **균 구매 소요 일수를 계산하고, 그 결과를** `user_data` **에 통합**

```
CREATE OR REPLACE TABLE project_name.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_name.modulabs_project.data
    WHERE CustomerID IS NOT NULL
```

```
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.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 valued-door-456102-k8.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS cancel_frequency
  FROM valued-door-456102-k8.modulabs_project.data
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  t.cancel_frequency AS cancel_rate
FROM valued-door-456102-k8.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

```sql
CREATE OR REPLACE TABLE valued-door-456102-k8.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS cancel_frequency
  FROM valued-door-456102-k8.modulabs_project.data
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  t.cancel_frequency AS cancel_rate
FROM valued-door-456102-k8.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

쿼리 결과

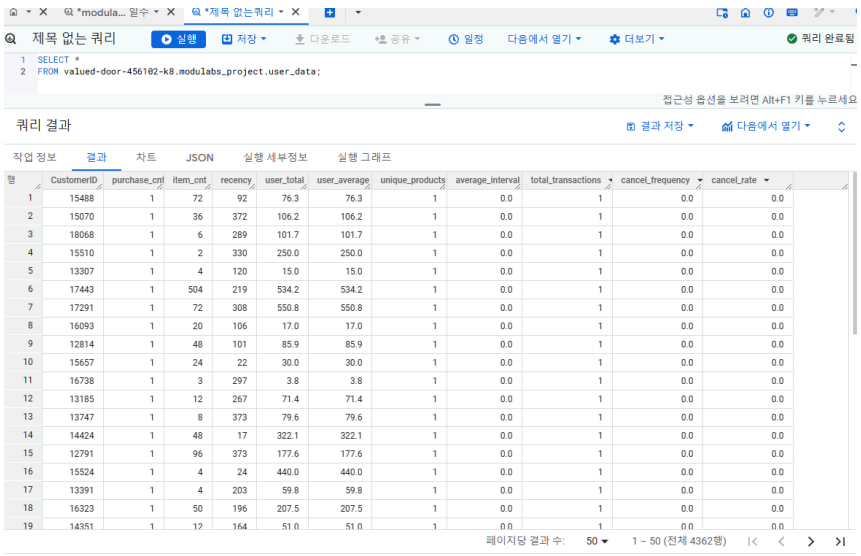
작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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

```sql
SELECT *
FROM valued-door-456102-k8.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]



제목 없는 쿼리 | 실행 | 저장 | 다운로드 | 공유 | 일정 | 다음에서 열기 | 더보기 | 쿼리 완료됨

```sql
SELECT *
FROM valued-door-456102-k8.modulabs_project.user_data;
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과 | 결과 저장 | 다음에서 열기

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	15488	1	72	92	76.3	76.3	1	0.0	1	0.0	0.0
2	15070	1	36	372	106.2	106.2	1	0.0	1	0.0	0.0
3	18068	1	6	289	101.7	101.7	1	0.0	1	0.0	0.0
4	15510	1	2	330	250.0	250.0	1	0.0	1	0.0	0.0
5	13307	1	4	120	15.0	15.0	1	0.0	1	0.0	0.0
6	17443	1	504	219	534.2	534.2	1	0.0	1	0.0	0.0
7	17291	1	72	308	550.8	550.8	1	0.0	1	0.0	0.0
8	16093	1	20	106	17.0	17.0	1	0.0	1	0.0	0.0
9	12814	1	48	101	85.9	85.9	1	0.0	1	0.0	0.0
10	15657	1	24	22	30.0	30.0	1	0.0	1	0.0	0.0
11	16738	1	3	297	3.8	3.8	1	0.0	1	0.0	0.0
12	13185	1	12	267	71.4	71.4	1	0.0	1	0.0	0.0
13	13747	1	8	373	79.6	79.6	1	0.0	1	0.0	0.0
14	14424	1	48	17	322.1	322.1	1	0.0	1	0.0	0.0
15	12791	1	96	373	177.6	177.6	1	0.0	1	0.0	0.0
16	15524	1	4	24	440.0	440.0	1	0.0	1	0.0	0.0
17	13391	1	4	203	59.8	59.8	1	0.0	1	0.0	0.0
18	16323	1	50	196	207.5	207.5	1	0.0	1	0.0	0.0
19	14351	1	12	164	51.0	51.0	1	0.0	1	0.0	0.0

페이지당 결과 수: 50 | 1 – 50 (전체 4362행)

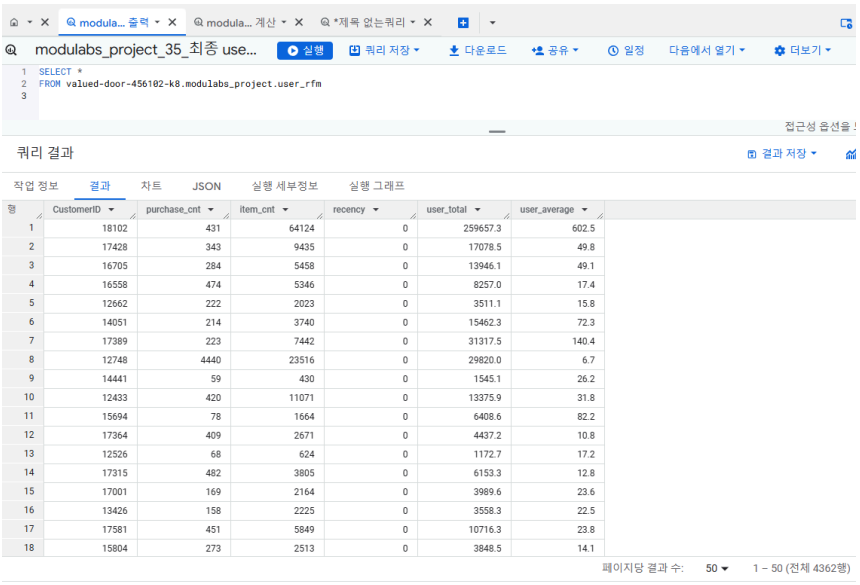
회고

[회고 내용을 작성해주세요]

Keep : 조금은 늦었지만 끝까지 오늘의 프로젝트를 완료한점이 무척 좋았습니다.

Problem : 개념이 많이 익숙하지 않아서 이것저것 시도해보느냐 많은 시간이 걸렸습니다.

Try : 다음에는 개념이 많이 익숙해져서 프로젝트를 시간내에 해결해보고 싶습니다.



modula... 출력 | modula... 계산 | *제목 없는쿼리

modulabs_project_35_최종 use... 실행 | 쿼리 저장 | 다운로드 | 공유 | 일정 | 다음에서 열기 | 더보기

```
SELECT *
FROM valued-door-456102-k8.modulabs_project.user_rfm
```

접근성 옵션을

쿼리 결과 | 결과 저장

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	18102	431	64124	0	259657.3	602.5
2	17428	343	9435	0	17078.5	49.8
3	16705	284	5458	0	13946.1	49.1
4	16558	474	5346	0	8257.0	17.4
5	12662	222	2023	0	3511.1	15.8
6	14051	214	3740	0	15462.3	72.3
7	17389	223	7442	0	31317.5	140.4
8	12748	4440	23516	0	29820.0	6.7
9	14441	59	430	0	1545.1	26.2
10	12433	420	11071	0	13375.9	31.8
11	15694	78	1664	0	6408.6	82.2
12	17364	409	2671	0	4437.2	10.8
13	12526	68	624	0	1172.7	17.2
14	17315	482	3805	0	6153.3	12.8
15	17001	169	2164	0	3989.6	23.6
16	13426	158	2225	0	3558.3	22.5
17	17581	451	5849	0	10716.3	23.8
18	15804	273	2513	0	3848.5	14.1

페이지당 결과 수: 50 1 – 50 (전체 4362행)