고객을 세그먼테이션하자 [프로젝트]

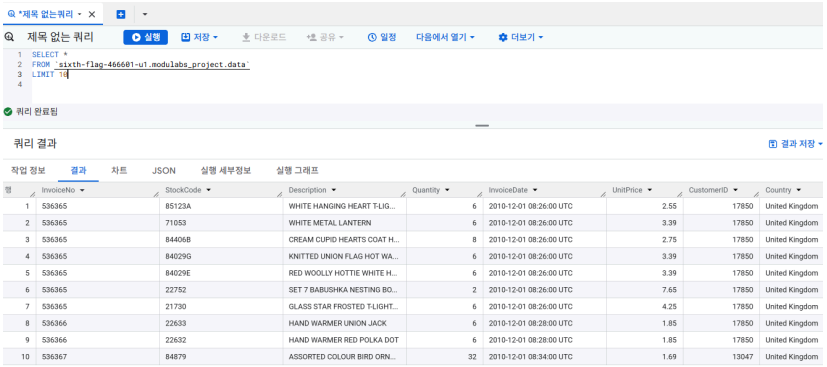
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
# [[YOUR QUERY]]
SELECT *
FROM `sixth-flag-466601-u1.modulabs_project.data`
LIMIT 10
```

[결과 이미지를 넣어주세요]

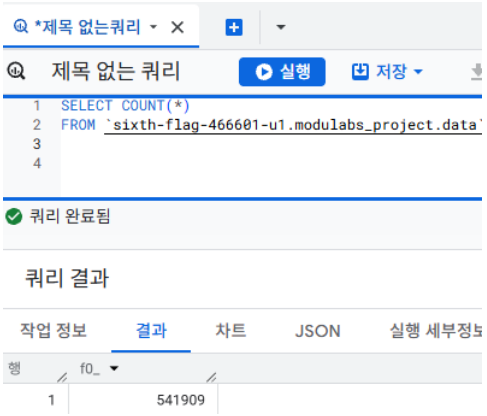


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
# [[YOUR QUERY]]
SELECT COUNT(*)
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

[결과 이미지를 넣어주세요]



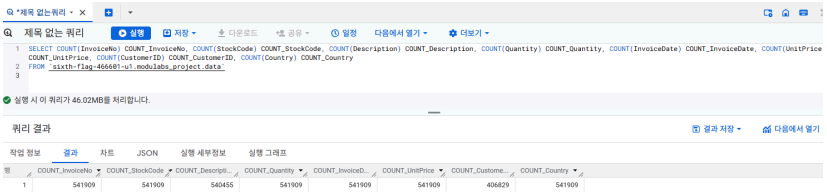
데이터 수 세기

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

```
# [[YOUR QUERY]]
SELECT COUNT(InvoiceNo) COUNT_InvoiceNo, COUNT(StockCode) COUNT_StockCode, COUNT(Description) COUNT_Descrip
```

```
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

[결과 이미지를 넣어주세요]



11-4. 데이터 전처리 방법(1): 결측치 제거

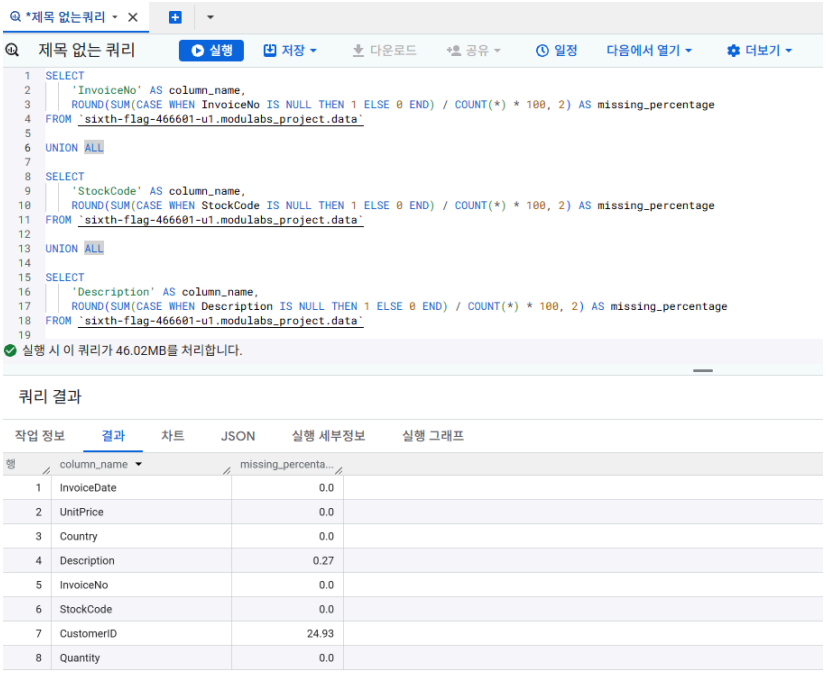
컬럼 별 누락된 값의 비율 계산

- **각 컬럼 별 누락된 값의 비율을 계산**
  - **각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기**

```
# [[YOUR QUERY]]
SELECT
   'InvoiceNo' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `sixth-flag-466601-u1.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_percentage
FROM `sixth-flag-466601-u1.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_percentage
FROM `sixth-flag-466601-u1.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_percentage
FROM `sixth-flag-466601-u1.modulabs_project.data`

UNION ALL

SELECT
   'InvoiceDate' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `sixth-flag-466601-u1.modulabs_project.data`

UNION ALL

SELECT
   'UnitPrice' AS column_name,
   ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `sixth-flag-466601-u1.modulabs_project.data`

UNION ALL
```

```
SELECT
   'CustomerID' AS column_name,
   ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `sixth-flag-466601-u1.modulabs_project.data`

UNION ALL

SELECT
   'Country' AS column_name,
   ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

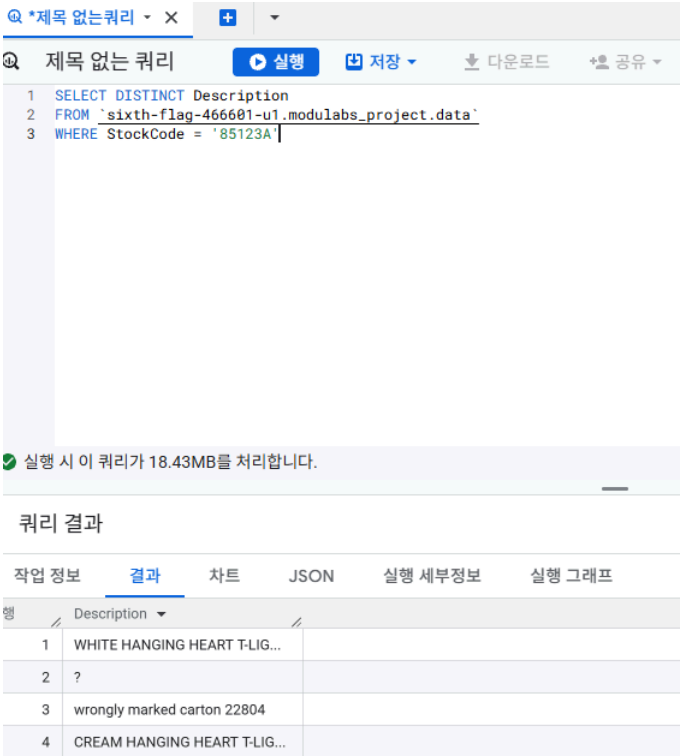


결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM project_name.modulabs_project.data
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

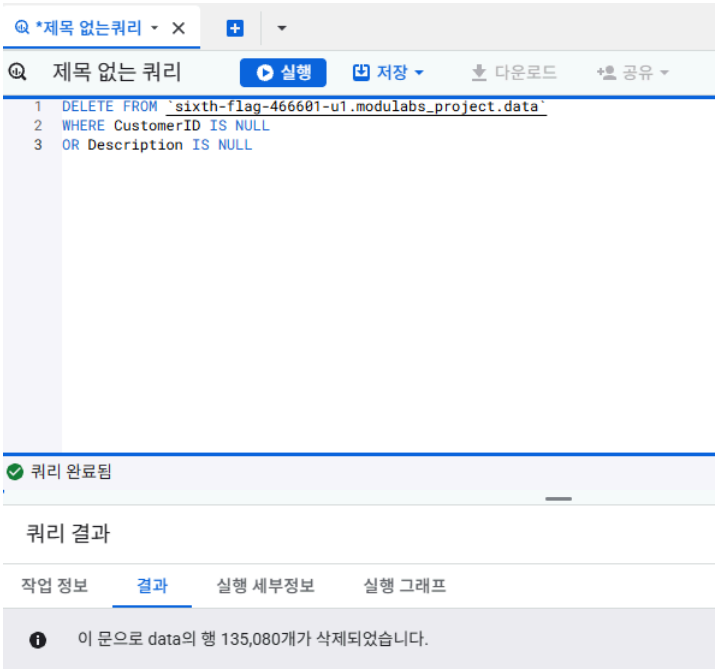


결측치 처리

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

```
DELETE FROM project_name.modulabs_project.data
WHERE CustomerID IS NULL
OR Description IS NULL;
```

[결과 이미지를 넣어주세요]



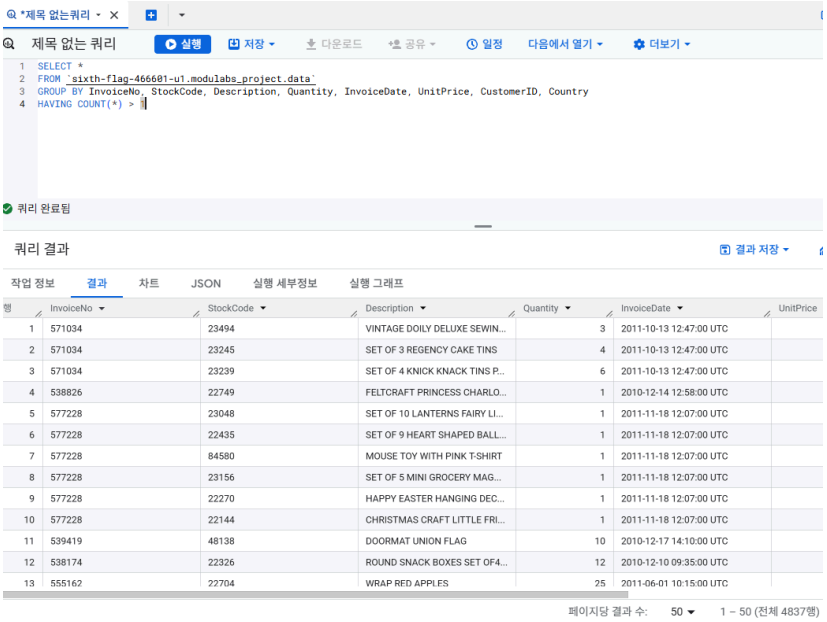
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT *
FROM project_name.modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING COUNT(*) > 1
```

[결과 이미지를 넣어주세요]

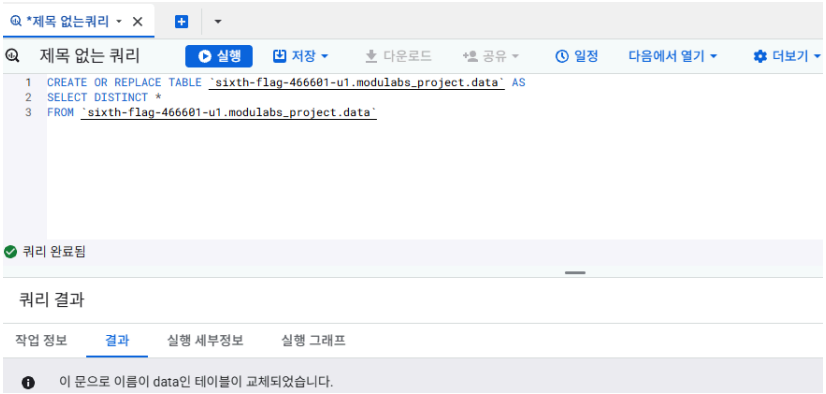


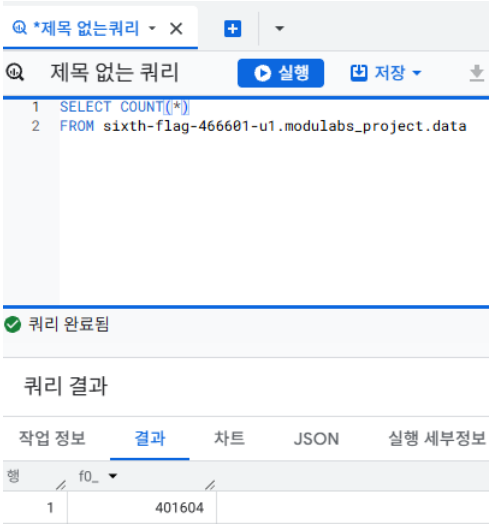
중복값 처리

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

```
# [[YOUR QUERY]];
CREATE OR REPLACE TABLE `sixth-flag-466601-u1.modulabs_project.data` AS
SELECT DISTINCT *
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

[결과 이미지를 넣어주세요]





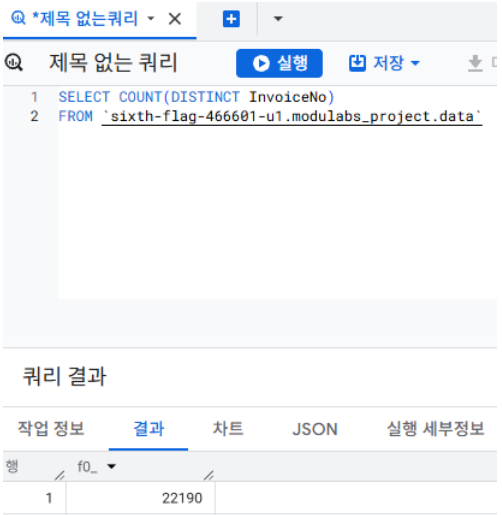
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT InvoiceNo)
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

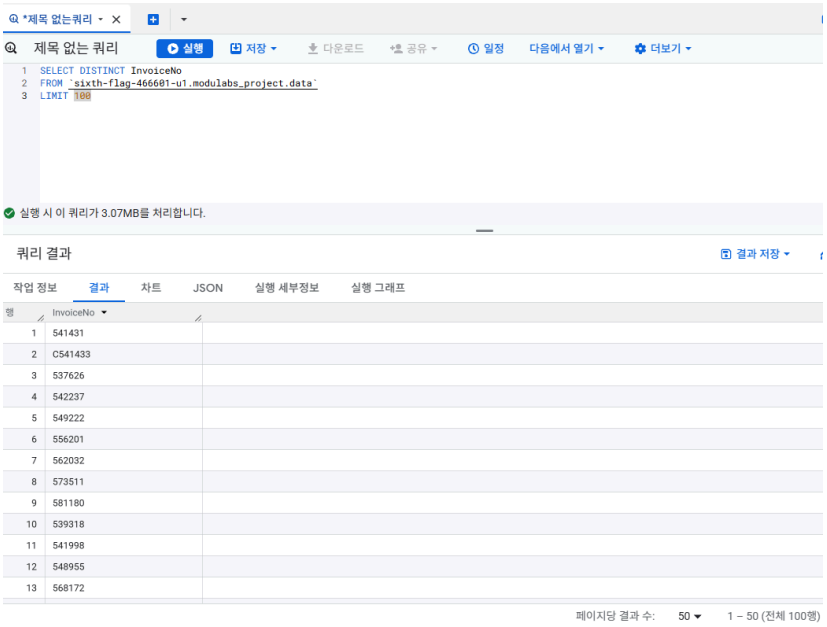
[결과 이미지를 넣어주세요]



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

```
# [[YOUR QUERY]]
SELECT DISTINCT InvoiceNo
FROM `sixth-flag-466601-u1.modulabs_project.data`
LIMIT 100
```

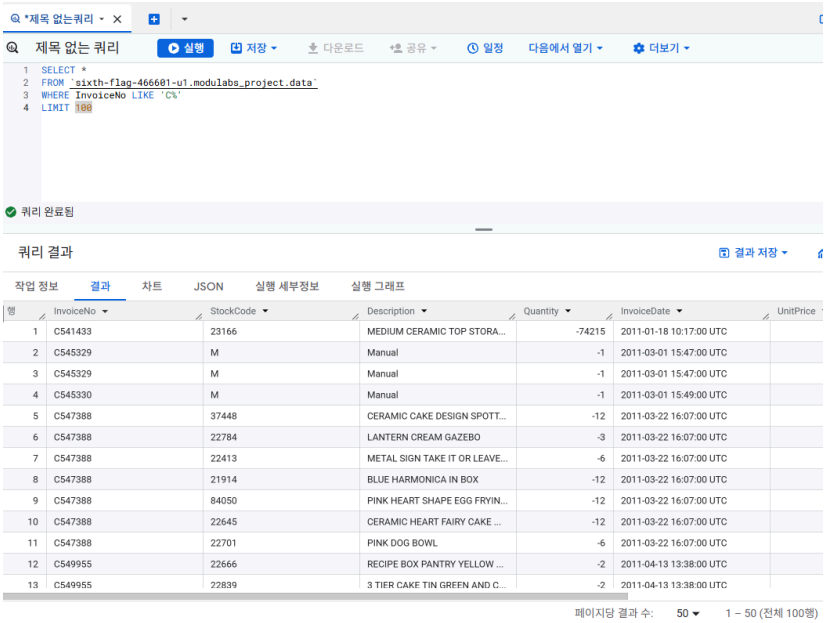
[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

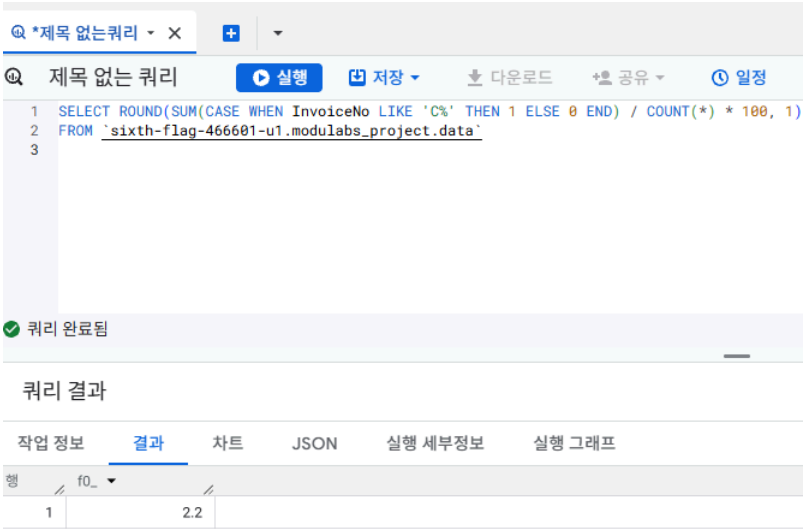
[결과 이미지를 넣어주세요]



- **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)
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

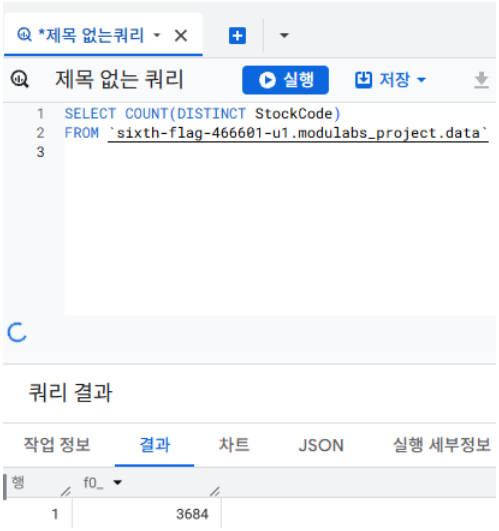


`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT StockCode)
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

[결과 이미지를 넣어주세요]



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

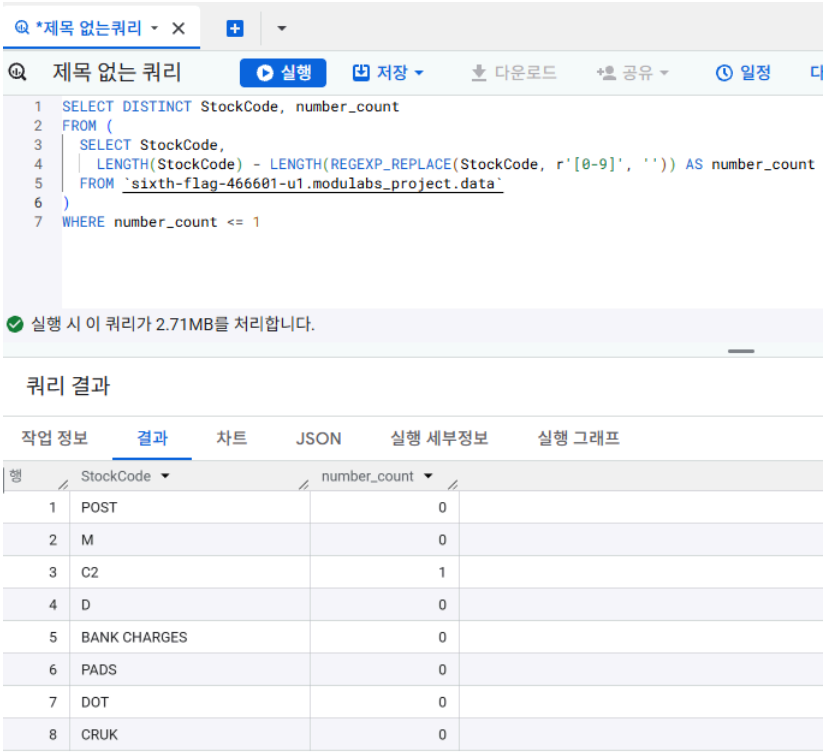
```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM project_name.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

[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 project_name.modulabs_project.data
)
WHERE number_count <= 1
```

[결과 이미지를 넣어주세요]



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)

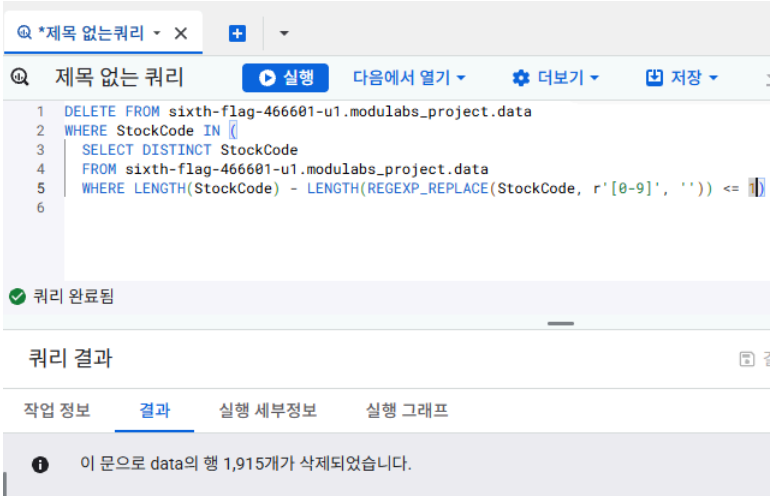
```
SELECT ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM (SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `sixth-flag-466601-u1.modulabs_project.data`)
```

[결과 이미지를 넣어주세요]

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM project_name.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM sixth-flag-466601-u1.modulabs_project.data
  WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1)
```

[결과 이미지를 넣어주세요]

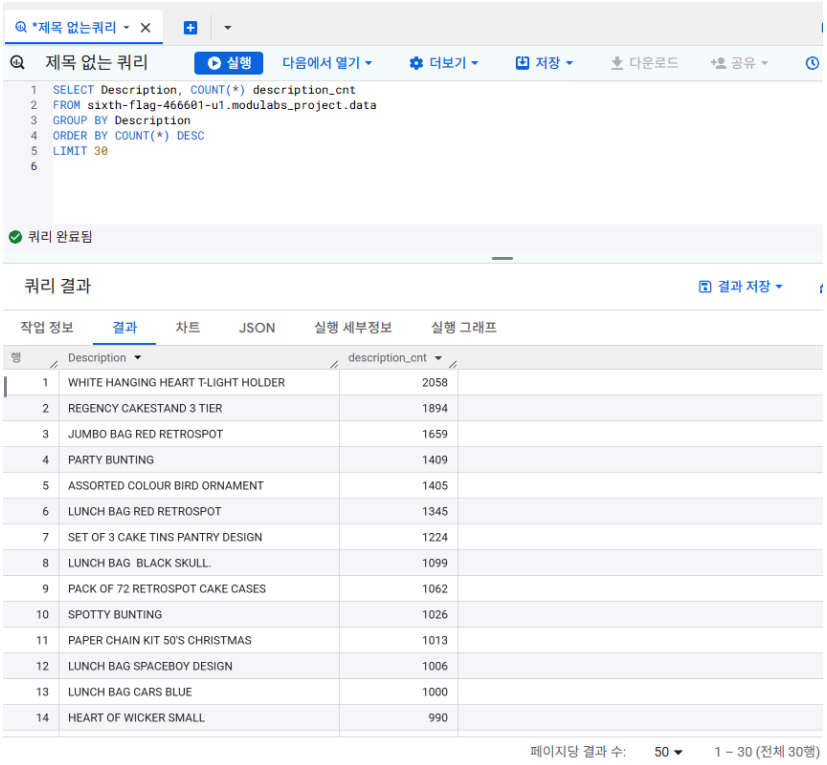


Description 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM project_name.modulabs_project.data
GROUP BY Description
ORDER BY COUNT(*) DESC
LIMIT 30
```

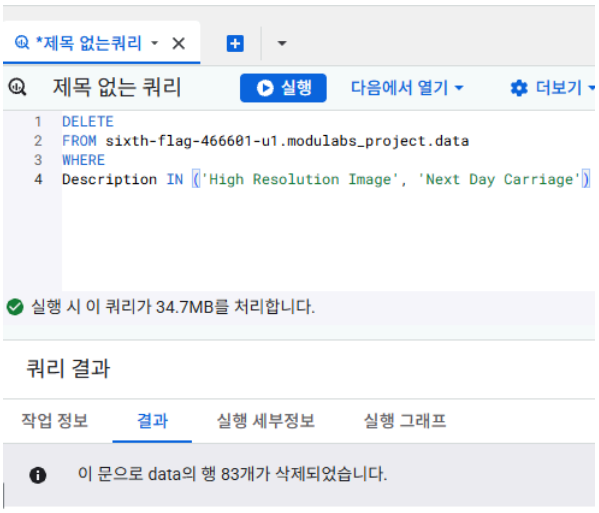
[결과 이미지를 넣어주세요]



- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM project_name.modulabs_project.data
WHERE
Description IN ('High Resolution Image', 'Next Day Carriage')
```

[결과 이미지를 넣어주세요]

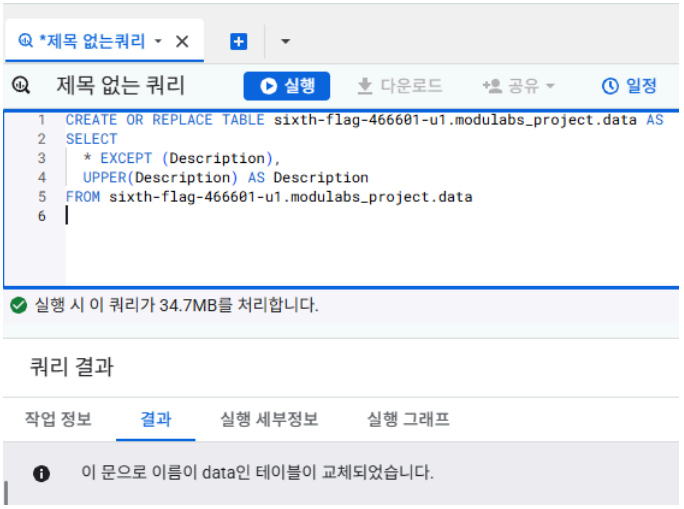


- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT
```

```
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

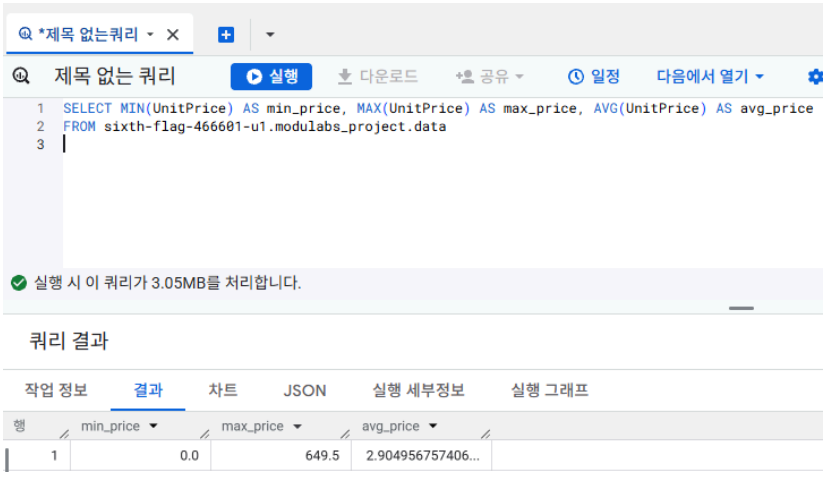


UnitPrice 살펴보기

- **UnitPrice 의 최솟값, 최댓값, 평균을 구하기**

```
SELECT SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM project_name.modulabs_project.data;
```

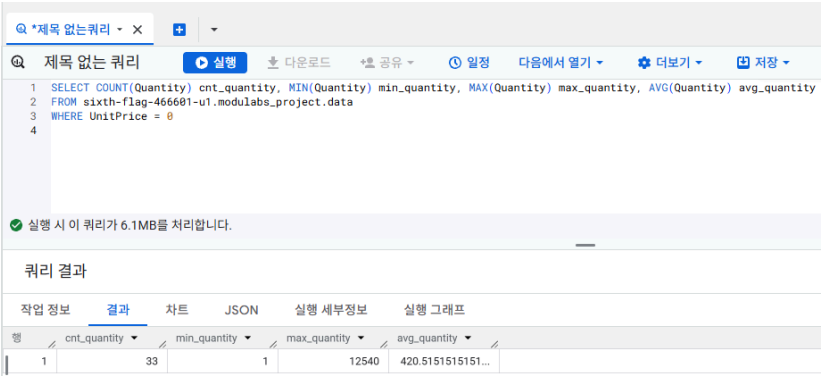
[결과 이미지를 넣어주세요]



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

```
SELECT COUNT(Quantity) cnt_quantity, MIN(Quantity) min_quantity, MAX(Quantity) max_quantity, AVG(Quantity) avg_quantity
FROM project_name.modulabs_project.data
WHERE UnitPrice = 0
```

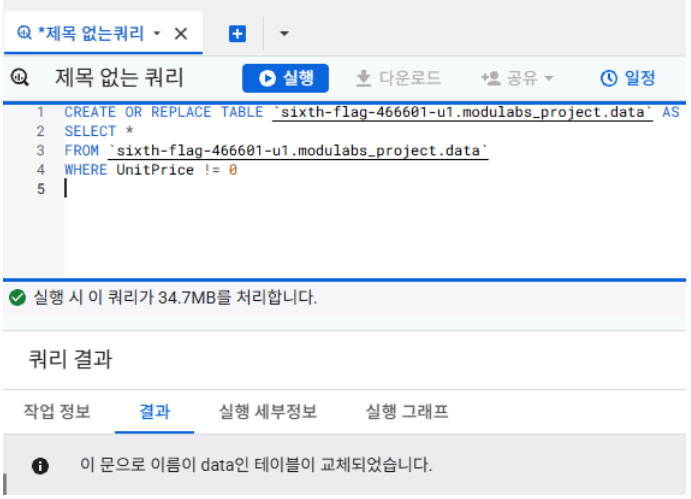
[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT *
FROM project_name.modulabs_project.data
WHERE UnitPrice != 0
```

[결과 이미지를 넣어주세요]



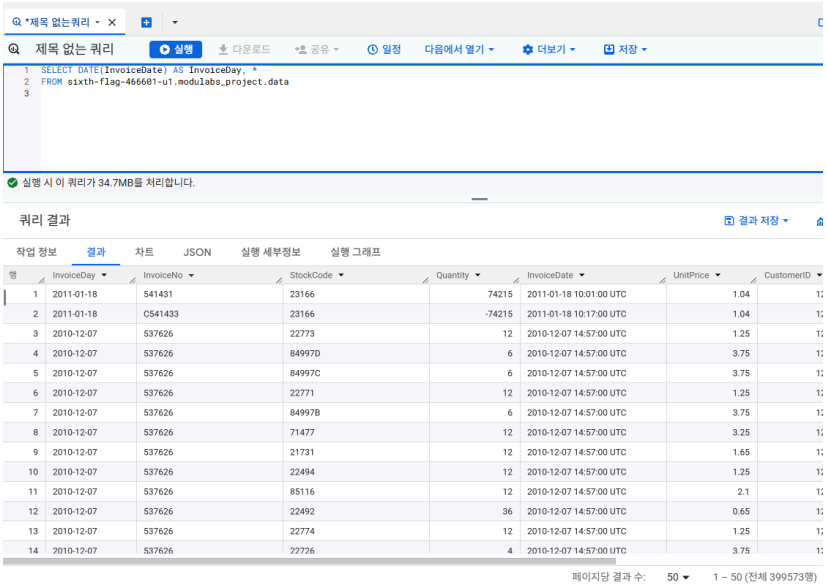
11-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```

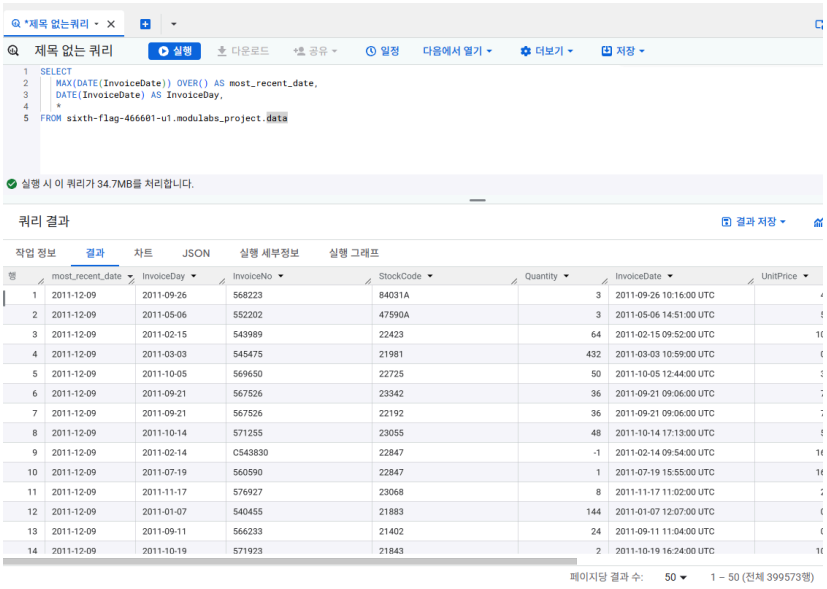
[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(DATE(InvoiceDate)) OVER() AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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

```
SELECT
   CustomerID,
   MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM project_name.modulabs_project.data
GROUP BY CustomerID
```

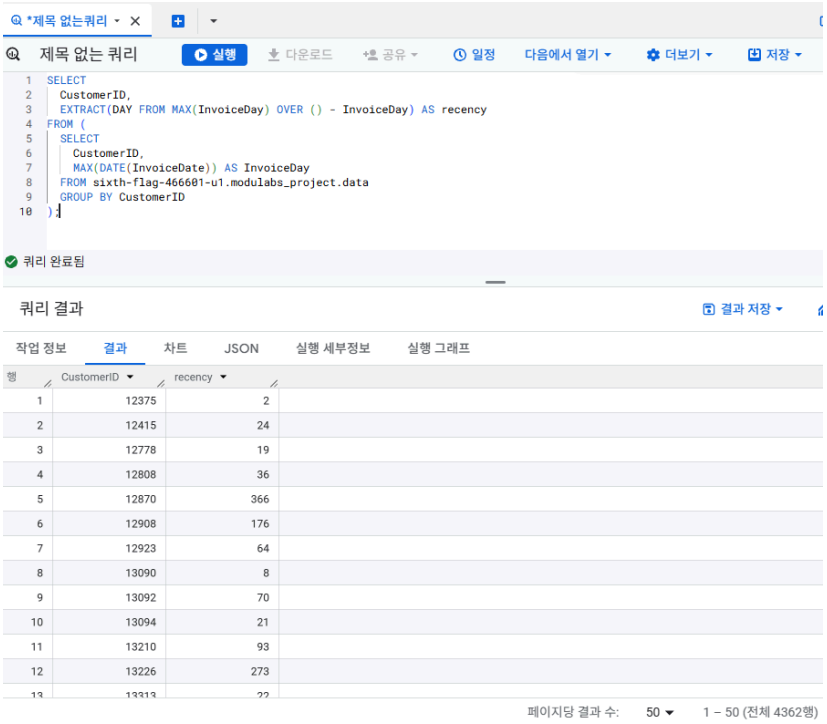
[결과 이미지를 넣어주세요]

행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17
11	12357	2011-11-06
12	12358	2011-12-08
13	12359	2011-12-02
14	12360	2011-10-18

- **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]



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

```
CREATE OR REPLACE TABLE project_name.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 `sixth-flag-466601-u1.modulabs_project.data`
  GROUP BY CustomerID
)
```

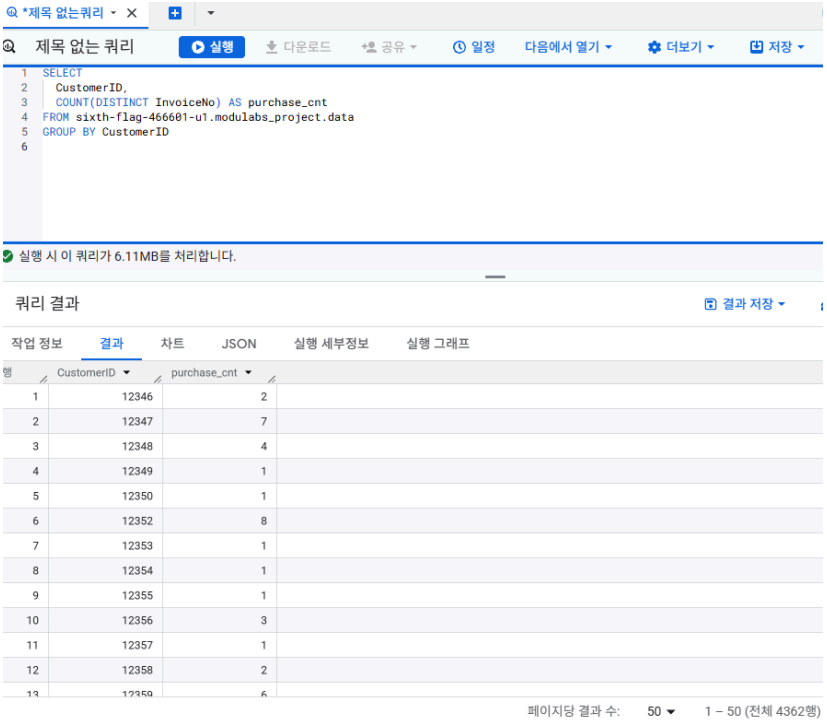
[결과 이미지를 넣어주세요]

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM project_name.modulabs_project.data
GROUP BY CustomerID
```

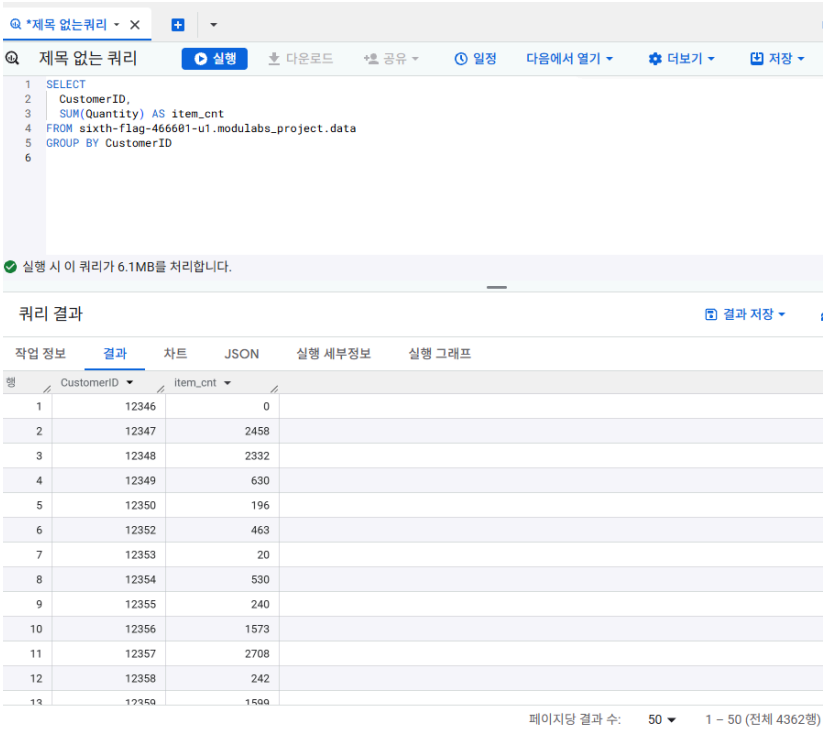
[결과 이미지를 넣어주세요]



- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM project_name.modulabs_project.data
GROUP BY CustomerID
```

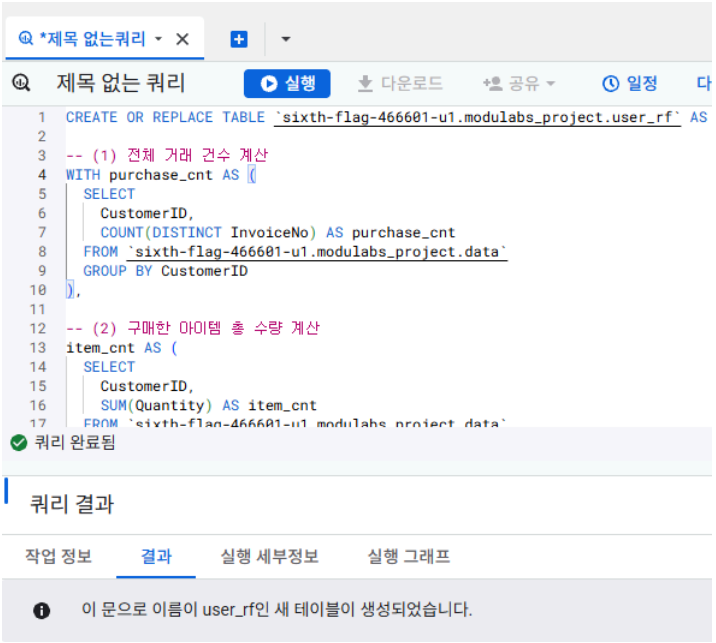
[결과 이미지를 넣어주세요]

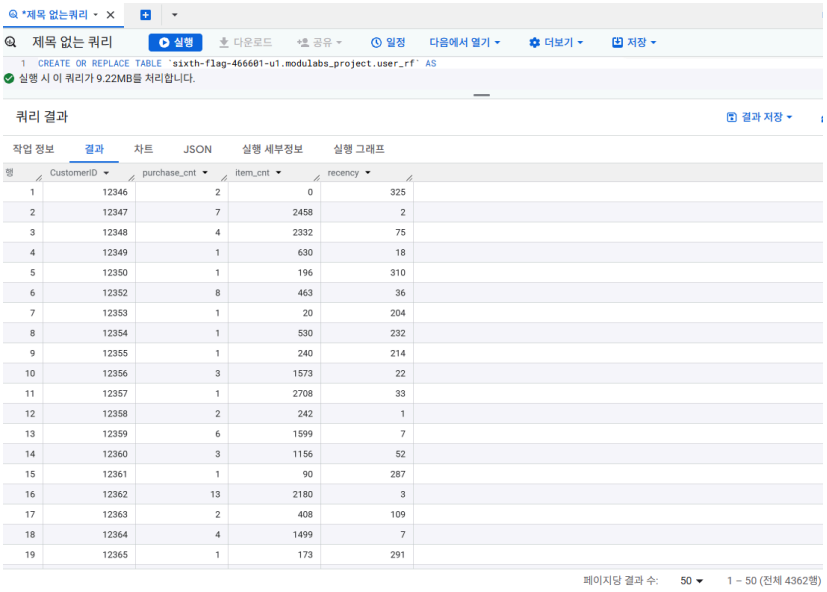


- **전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 `user_rf` 라는 이름의 테이블에 저장하기**

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
),

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

[결과 이미지를 넣어주세요]



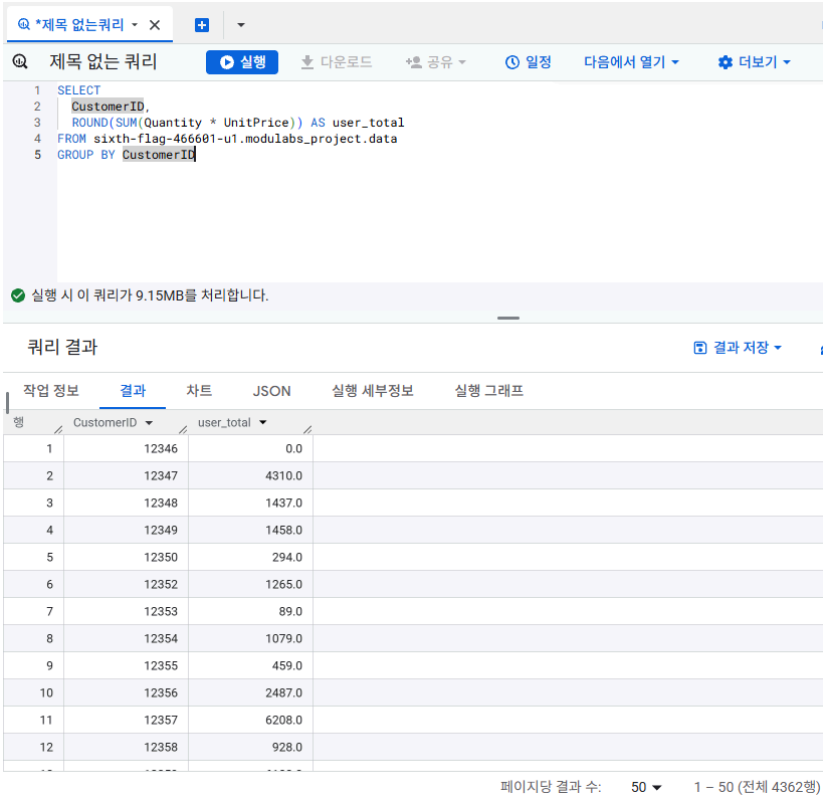


Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice)) AS user_total
FROM project_name.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 project_name.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ut.user_total / rf.purchase_cnt AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID
```

[결과 이미지를 넣어주세요]

*제목 없는쿼리

제목 없는 쿼리 실행 다운로드 공유 일정

```
CREATE OR REPLACE TABLE sixth-flag-466601-u1.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ut.user_total / rf.purchase_cnt AS user_average
FROM sixth-flag-466601-u1.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM sixth-flag-466601-u1.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID
```

쿼리 완료됨

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

user_rfm 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13298	1	96	1	360.0	360.0
3	14569	1	79	1	227.0	227.0
4	15520	1	314	1	343.0	343.0
5	13436	1	76	1	197.0	197.0
6	15471	1	256	2	454.0	454.0
7	14204	1	72	2	151.0	151.0
8	15195	1	1404	2	3861.0	3861.0
9	16569	1	93	3	124.0	124.0
10	17914	1	457	3	329.0	329.0
11	14578	1	240	3	169.0	169.0
12	15992	1	17	3	42.0	42.0
13	12478	1	233	3	546.0	546.0
14	12650	1	250	3	242.0	242.0
15	16528	1	171	3	244.0	244.0
16	12442	1	181	3	144.0	144.0
17	15318	1	642	3	313.0	313.0
18	15097	1	170	4	248.0	248.0
19	12367	1	172	4	151.0	151.0
20	13790	1	748	4	349.0	349.0
21	17383	1	148	4	193.0	193.0
22	18015	1	157	4	120.0	120.0
23	16597	1	184	4	90.0	90.0

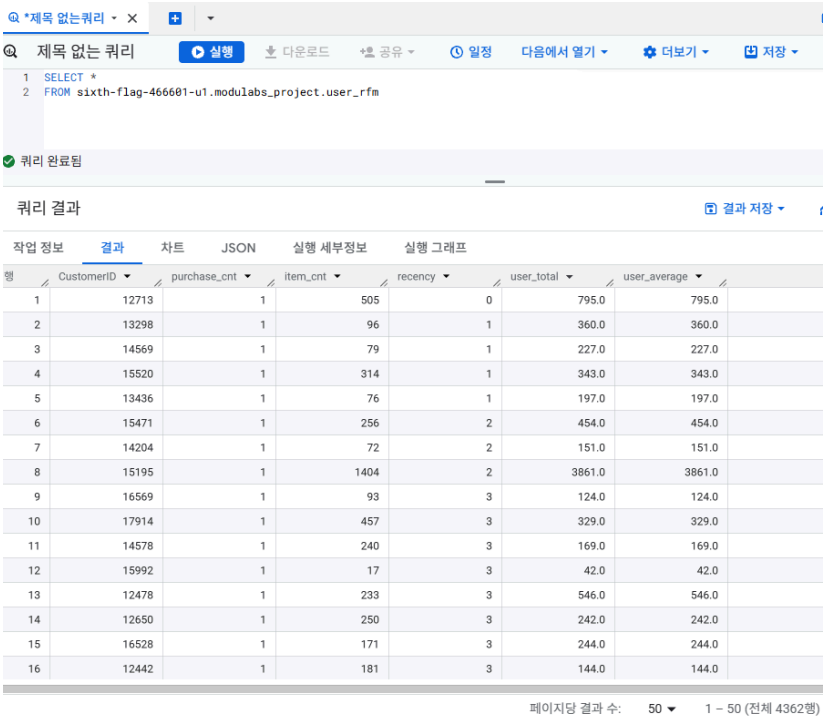
페이지당 결과 수: 50 1 – 50 (전체 4362행)

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM sixth-flag-466601-u1.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13298	1	96	1	360.0	360.0
3	14569	1	79	1	227.0	227.0
4	15520	1	314	1	343.0	343.0
5	13436	1	76	1	197.0	197.0
6	15471	1	256	2	454.0	454.0
7	14204	1	72	2	151.0	151.0
8	15195	1	1404	2	3861.0	3861.0
9	16569	1	93	3	124.0	124.0
10	17914	1	457	3	329.0	329.0
11	14578	1	240	3	169.0	169.0
12	15992	1	17	3	42.0	42.0
13	12478	1	233	3	546.0	546.0
14	12650	1	250	3	242.0	242.0
15	16528	1	171	3	244.0	244.0
16	12442	1	181	3	144.0	144.0

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]

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	12713	1	505	0	795.0	795.0	37
2	13298	1	96	1	360.0	360.0	2
3	14569	1	79	1	227.0	227.0	10
4	15520	1	314	1	343.0	343.0	18
5	13436	1	76	1	197.0	197.0	12
6	15471	1	256	2	454.0	454.0	67
7	14204	1	72	2	151.0	151.0	36
8	15195	1	1404	2	3861.0	3861.0	1
9	16569	1	93	3	124.0	124.0	5
10	17914	1	457	3	329.0	329.0	72
11	14578	1	240	3	169.0	169.0	24
12	15992	1	17	3	42.0	42.0	3
13	12478	1	233	3	546.0	546.0	35
14	12650	1	250	3	242.0	242.0	19
15	16528	1	171	3	244.0	244.0	17
16	12442	1	181	3	144.0	144.0	11
17	15318	1	642	3	313.0	313.0	33
18	15097	1	170	4	248.0	248.0	25

페이지당 결과 수: 50 1 – 50 (전체 4362행)

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]

user_data

쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...
51	15940	1	4	311	36.0	36.0	1	0.0
52	13366	1	144	50	56.0	56.0	1	0.0
53	18174	1	50	7	104.0	104.0	1	0.0
54	16144	1	16	246	175.0	175.0	1	0.0
55	17291	1	72	308	551.0	551.0	1	0.0
56	18113	1	72	368	76.0	76.0	1	0.0
57	16078	1	16	283	79.0	79.0	1	0.0
58	15316	1	100	326	165.0	165.0	1	0.0
59	16990	1	100	218	179.0	179.0	1	0.0
60	18133	1	1350	212	931.0	931.0	1	0.0
61	14119	1	-2	354	-20.0	-20.0	1	0.0
62	17307	1	-144	365	-153.0	-153.0	1	0.0
63	14705	1	100	198	179.0	179.0	1	0.0
64	15668	1	72	217	76.0	76.0	1	0.0
65	12943	1	-1	301	-4.0	-4.0	1	0.0
66	16148	1	72	296	76.0	76.0	1	0.0
67	18068	1	6	289	102.0	102.0	1	0.0
68	16323	1	50	196	208.0	208.0	1	0.0
69	13017	1	48	7	204.0	204.0	1	0.0
70	16428	1	-1	81	-3.0	-3.0	1	0.0
71	16257	1	1	176	22.0	22.0	1	0.0
72	14576	1	12	372	35.0	35.0	1	0.0
73	15070	1	36	372	106.0	106.0	1	0.0

페이지당 결과 수: 50 51 – 100 (전체 4362행)

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 sixth-flag-466601-u1.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNTIF(DISTINCT (InvoiceNo LIKE 'C%')) AS cancel_frequency
  FROM sixth-flag-466601-u1.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_transactions, 2) cancel_rate
FROM `sixth-flag-466601-u1.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
```

[결과 이미지를 넣어주세요]

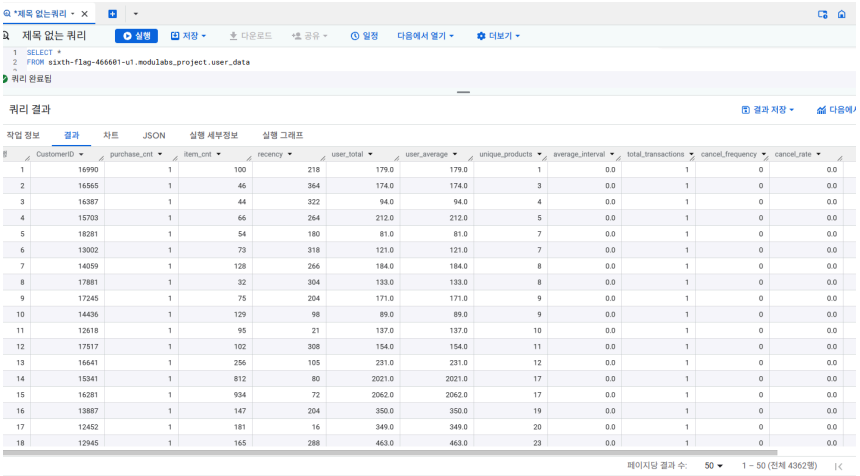
user_data

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...	total_transac...	cancel_frequ...	cancel_rate
1	16990	1	100	218	179.0	179.0	1	0.0	1	0	0.0
2	16565	1	46	364	174.0	174.0	3	0.0	1	0	0.0
3	16387	1	44	322	94.0	94.0	4	0.0	1	0	0.0
4	15703	1	66	264	212.0	212.0	5	0.0	1	0	0.0
5	18281	1	54	180	81.0	81.0	7	0.0	1	0	0.0
6	13002	1	73	318	121.0	121.0	7	0.0	1	0	0.0
7	14059	1	128	266	184.0	184.0	8	0.0	1	0	0.0
8	17881	1	32	304	133.0	133.0	8	0.0	1	0	0.0
9	17245	1	75	204	171.0	171.0	9	0.0	1	0	0.0
10	14436	1	129	98	89.0	89.0	9	0.0	1	0	0.0
11	12618	1	95	21	137.0	137.0	10	0.0	1	0	0.0
12	17517	1	102	308	154.0	154.0	11	0.0	1	0	0.0
13	16641	1	256	105	231.0	231.0	12	0.0	1	0	0.0
14	15341	1	812	80	2021.0	2021.0	17	0.0	1	0	0.0
15	16281	1	934	72	2062.0	2062.0	17	0.0	1	0	0.0
16	13887	1	147	204	350.0	350.0	19	0.0	1	0	0.0
17	12452	1	181	16	349.0	349.0	20	0.0	1	0	0.0
18	12945	1	165	288	463.0	463.0	23	0.0	1	0	0.0
19	17263	1	36	208	63.0	63.0	23	0.0	1	0	0.0
20	15592	1	354	46	389.0	389.0	24	0.0	1	0	0.0
21	14873	1	168	210	520.0	520.0	28	0.0	1	0	0.0
22	12638	1	573	33	512.0	512.0	38	0.0	1	0	0.0

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

```
SELECT *
FROM sixth-flag-466601-u1.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16990	1	100	218	179.0	179.0	1	0.0	1	0	0.0
2	16565	1	46	364	174.0	174.0	3	0.0	1	0	0.0
3	16387	1	44	322	94.0	94.0	4	0.0	1	0	0.0
4	15703	1	66	264	212.0	212.0	5	0.0	1	0	0.0
5	18281	1	54	180	81.0	81.0	7	0.0	1	0	0.0
6	13002	1	73	318	121.0	121.0	7	0.0	1	0	0.0
7	14059	1	128	266	184.0	184.0	8	0.0	1	0	0.0
8	17881	1	32	304	133.0	133.0	8	0.0	1	0	0.0
9	17245	1	75	204	171.0	171.0	9	0.0	1	0	0.0
10	14436	1	129	98	89.0	89.0	9	0.0	1	0	0.0
11	12618	1	95	21	137.0	137.0	10	0.0	1	0	0.0
12	17517	1	102	308	154.0	154.0	11	0.0	1	0	0.0
13	16641	1	256	105	231.0	231.0	12	0.0	1	0	0.0
14	15341	1	812	80	2021.0	2021.0	17	0.0	1	0	0.0
15	16281	1	934	72	2062.0	2062.0	17	0.0	1	0	0.0
16	13887	1	147	204	350.0	350.0	19	0.0	1	0	0.0
17	12452	1	181	16	349.0	349.0	20	0.0	1	0	0.0
18	12945	1	165	288	463.0	463.0	23	0.0	1	0	0.0

데이터 해석 및 통찰

CustomerID의 경우 결측치 비율이 24.93%로 높은 수준임에도 노이즈가 될 수 있기에 다른 값으로 대체하는 대신 제거하는 것을 통해 분석 목적에 따라 불필요한 데이터는 과감하게 제거해야 한다는 것을 확인할 수 있었으며 데이터 전처리 중 오류값 처리에 가장 많은 시간과 노력이 든다는 것을 직접 깨달을 수 있었습니다.

쿼리 결과

작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

행	country	cnt
1	United Kingdom	356008
2	Germany	9079
3	France	8152
4	EIRE	7368
5	Spain	2462
6	Netherlands	2326
7	Belgium	1971
8	Switzerland	1843
9	Portugal	1427
10	Australia	1253
11	Norway	1059
12	Italy	783
13	Channel Islands	752
14	Finland	653
15	Cyprus	608
16	Sweden	436

데이터 수를 국가별로 내림차순 정렬해보니, 영국이 2등 국가인 독일에 비해 약 40배정도 데이터가 많은 것을 확인할 수 있었으며, 영국 고객이 상당히 많다는 것을 고려하여 영국인들의 취향을 고려한 맞춤형 상품을 내놓게 된다면 매출 증가를 가져올 수 있을 것 같아 보였습니다.

쿼리 결과

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

행	Description	cnt
1	WHITE HANGING HEART T-LIGHT HOLDER	2058
2	REGENCY CAKESTAND 3 TIER	1893
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1408
5	ASSORTED COLOUR BIRD ORNAMENT	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY DESIGN	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE CASES	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRISTMAS	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000

LIGHT HOLDER, CAKESTAND, JUMBO BAG 등 가정용품, 주방용품, 장식품 등의 판매 수량이 높았으며 이에 대한 추가 재고 확보 및 다양한 아이템 확보에 신경을 쓰는 것이 좋을 것 같아 보였습니다.

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

```
SELECT *
FROM sixth-flag-466601-u1.modulabs_project.user_data
WHERE cancel_rate >= 0.1
ORDER BY cancel_rate DESC
```

쿼리 완료됨

쿼리 결과 | 결과 저장

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

user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
-135.0	4	0.0	1	1	1.0
-20.0	1	0.0	1	1	1.0
-227.0	2	0.0	1	1	1.0
-153.0	1	0.0	1	1	1.0
-4.0	1	0.0	1	1	1.0
-812.0	15	0.0	1	1	1.0
-102.0	1	0.0	1	1	1.0
-465.0	3	0.0	1	1	1.0
-3.0	1	0.0	1	1	1.0
-102.0	5	0.0	1	1	1.0
-8.0	3	0.0	1	1	1.0
-32.0	3	0.0	1	1	1.0
-35.0	1	0.0	1	1	1.0
-12.0	2	0.0	1	1	1.0
-31.0	1	0.0	1	1	1.0
-94.0	2	0.0	1	1	1.0

페이지당 결과 수: 50 | 1 – 50 (전체 1135행)

10건 중 1건 취소부터는 고려해 볼 만하다고 생각되어 기준을 0.1로 잡아 확인해보니, 1135명(비율로는 약 25%)이 해당됐으며, 고객들이 취소한 상품이 무엇인지, 왜 취소하는지를 생각해볼 필요가 있을 것 같습니다.

행	StockCode	Description	cancel_stock
1	22423	REGENCY CAKESTAND 3 TIER	180
2	22960	JAM MAKING SET WITH JARS	86
3	22720	SET OF 3 CAKE TINS PANTRY DESIGN	72
4	21232	STRAWBERRY CERAMIC TRINKET BOX	54
5	22699	ROSES REGENCY TEACUP AND SAUCER	53
6	22666	RECIPE BOX PANTRY YELLOW DESIGN	47
7	85099B	JUMBO BAG RED RETROSPOT	44
8	22697	GREEN REGENCY TEACUP AND SAUCER	42
9	20725	LUNCH BAG RED RETROSPOT	42
10	82483	WOOD 2 DRAWER CABINET WHITE FINISH	42
11	85123A	WHITE HANGING HEART T-LIGHT HOLDER	42
12	21843	RED RETROSPOT CAKE STAND	41
13	21314	SMALL GLASS HEART TRINKET POT	40
14	23245	SET OF 3 REGENCY CAKE TINS	37
15	22197	POPCORN HOLDER	36
16	22698	PINK REGENCY TEACUP AND SAUCER	35

페이지당 결과 수: 50

취소 수량이 많으 상품을 조사해보니, 판매량이 많았던 케이크틀, 가방 등이 보이는걸 봐서 많이 구매한 만큼 취소도 많이 되는 것을 확인할 수 있었고 리뷰나 취소 신청 이유 등을 참고하여 판매량에 비해 취소비율을 줄일 수 있는 방법을 고안해봐야 할 것 같습니다.

회고

Keep :

기존에 많이 연습하고, 사용했던 SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, LIMIT, WINDOW 함수 등의 기초 내용과 기본 사용 방법은 익숙해진 것 같습니다.

위 내용들만 안다고 해서 SQL의 모든 것을 아는 것은 아니기에 제가 더 편하게 구현하고자 했던 기능들이 따로 함수로 만들어져 있는지 검색해보며 새로운 내용을 익히는 능력도 키울 수 있었던 것 같습니다.

지금 알고 있는 내용을 반복 숙달하여 잊지 않도록 연습하는 것이 중요할 것 같습니다.

Problem :

SQL의 기본 문법들은 다 알고 있기에 단순히 이걸 적용만 하면 되지 않나 하고 생각했었습니다. 하지만 역시 도구의 사용 방법을 안다고 해도 이를 문제 해결에 어떻게 잘 적용할지는 또 다른 문제였던 것 같습니다.

특히 GROUP BY, 윈도우 함수 등을 응용하는 것에 있어서 좀 더 연습이 필요하다고 생각됩니다.

또한, COUNTIF, REGEXP 등 실무에서 유용하게 쓰일 수 있는 함수들에 대한 학습이 필요할 것 같습니다.

Try :

지금처럼 본인이 알고 있는 내용에 대해서는 검색하지 않고 스스로 해결하려고 노력하려 하고, 구현하고자 하지만 함수 사용법을 까먹어 모르는 것에 대해서는 적절하게 검색하여 알아보려는 태도를 가지는 것이 좋을 것 같습니다.

또한, 위 프로젝트를 해결했다고 해서 넘어갈게 아니라 반복 학습을 통해 위 프로젝트 내용이 쉽게 느껴지고, 가이드 없이도 수월하게 해결할 수 있도록 체득하는 것이 중요할 것 같습니다.