PDA

이커머스 고객 세그먼테이션 [프로젝트]

들어가며

A. 분석 목표

이커머스 플랫폼인 모두몰의 데이터 사이언티스트로서 데이터 분석을 진행하는 프로젝트.

모두몰에서는 고객별 맞춤 마케팅 전략을 새롭게 짜려는 프로젝트를 진행하고 있습니다. 이 프로젝트에 참여하는 여러분은 모두몰의 고객 데이터를 기반으로 고객별 구매 패턴을 파악하여 고객을 세그먼테이션해야 합니다.

B. 분석 방법론

고객별로 얼마나 최근에, 자주, 많은 금액을 지출했는지를 파악하여 고객을 나눌 수 있는 RFM(Recency, Frequency, Monetary) 분석 사용

C. 분석 프로세스

D. 데이터셋

UCI Machine Learning Repository에서 공개한 2010~2011 Kaggle E-Commerce Data

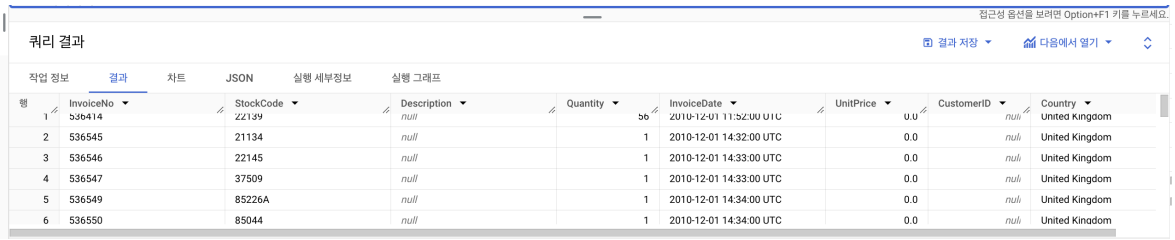
5-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
# [[YOUR QUERY]]
SELECT *
FROM modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

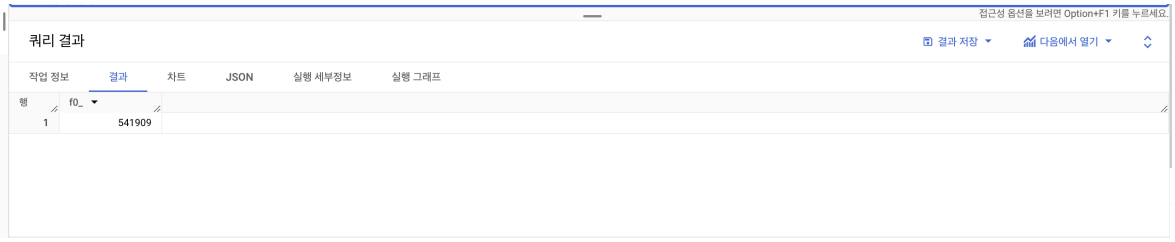


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
# [[YOUR QUERY]]
SELECT count(*)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



데이터 수 세기

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

```
# [[YOUR QUERY]]
SELECT count(InvoiceNo) AS COUNT_InvoiceNo,
       count(StockCode) AS COUNT_StockCodeNo,
       count(Description) AS COUNT_DescriptionNo,
       count(Quantity) AS COUNT_QuantityNo,
       count(InvoiceDate) AS COUNT_InvoiceDateNo,
       count(UnitPrice) AS COUNT_UnitPriceNo,
       count(CustomerID) AS COUNT_CustomerIDNo,
       count(Country) AS COUNT_CountryNo
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

5-4. 데이터 전처리 방법(1): 결측치 제거

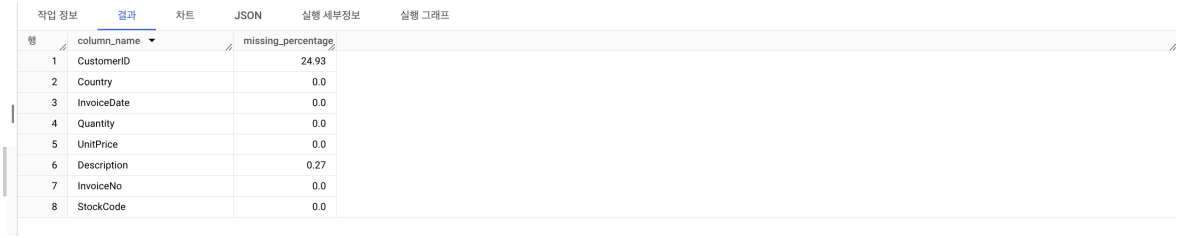
컬럼 별 누락된 값의 비율 계산

- **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(*
SELECT 'StockCode' AS column_name,ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*
SELECT 'Description' AS column_name,ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COU
SELECT 'Quantity' AS column_name,ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*)
```

```
SELECT 'InvoiceDate ' AS column_name,ROUND(SUM(CASE WHEN InvoiceDate  IS NULL THEN 1 ELSE 0 END) / C
SELECT 'UnitPrice' AS column_name,ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*
SELECT 'CustomerID' AS column_name,ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT
SELECT 'Country' AS column_name,ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) *
```

[결과 이미지를 넣어주세요]



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

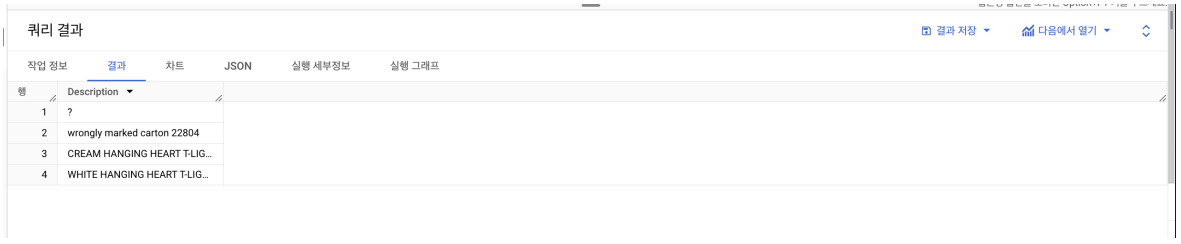
행	column_name	missing_percentage
1	CustomerID	24.93
2	Country	0.0
3	InvoiceDate	0.0
4	Quantity	0.0
5	UnitPrice	0.0
6	Description	0.27
7	InvoiceNo	0.0
8	StockCode	0.0

결측치 처리 전략

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

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

[결과 이미지를 넣어주세요]



쿼리 결과 | 결과 저장 | 다음에서 열기

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

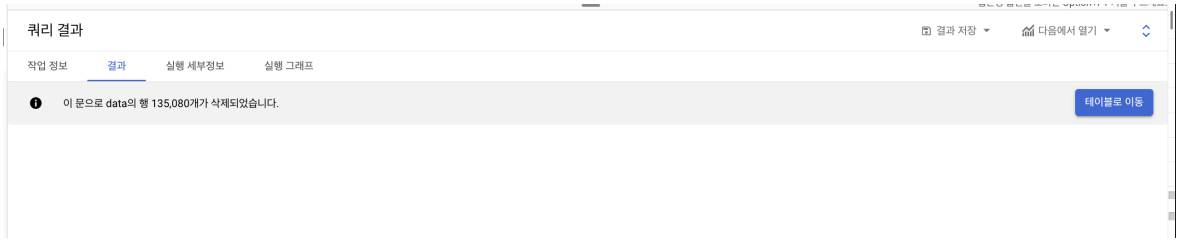
행	Description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	WHITE HANGING HEART T-LIG...

결측치 처리

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

```
DELETE FROM modulabs_project.data
WHERE Description is NULL OR CustomerID is null;
```

[결과 이미지를 넣어주세요]



쿼리 결과 | 결과 저장 | 다음에서 열기

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 data의 행 135,080개가 삭제되었습니다.

테이블로 이동

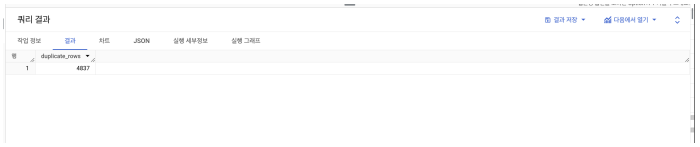
5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

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

[결과 이미지를 넣어주세요]

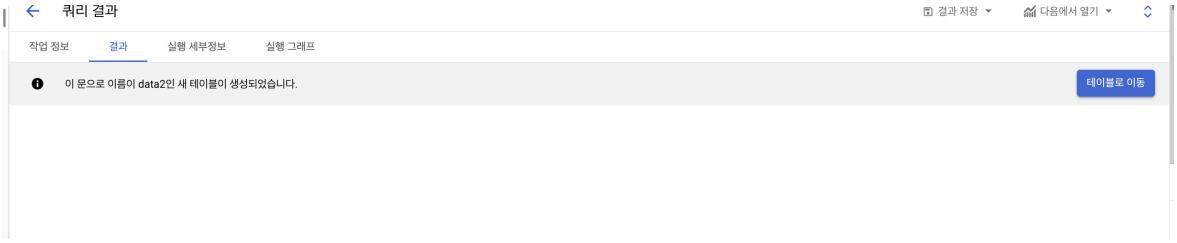


중복값 처리

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

```
# [[YOUR QUERY]];
CREATE OR REPLACE TABLE modulabs_project.data2 AS
SELECT DISTINCT * FROM `modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



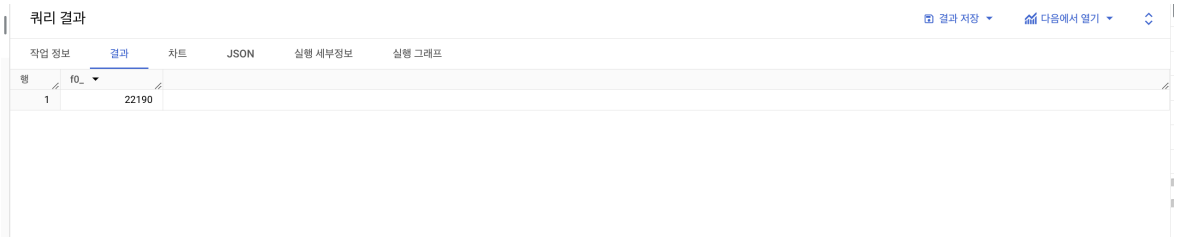
5-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

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

```
# [[YOUR QUERY]]
SELECT count(DISTINCT InvoiceNo)
FROM `modulabs_project.data2`;
```

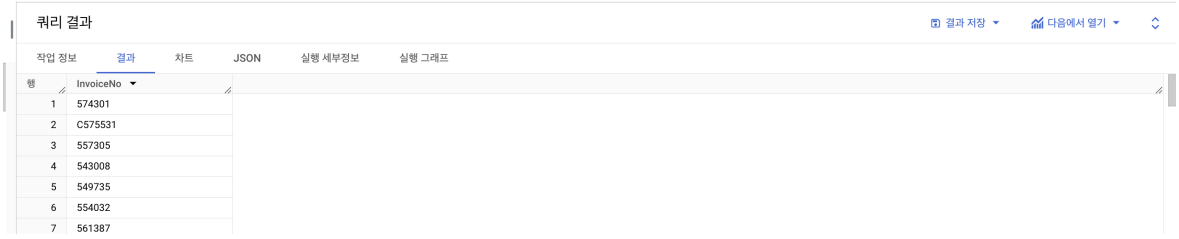
[결과 이미지를 넣어주세요]



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

```
# [[YOUR QUERY]]
SELECT DISTINCT InvoiceNo
FROM `modulabs_project.data2`
LIMIT 100;
```

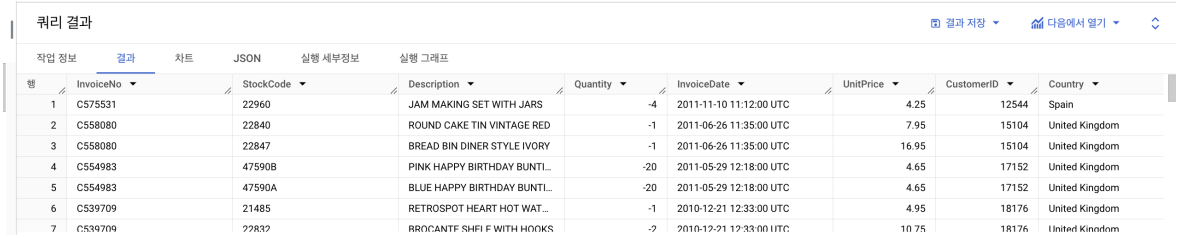
[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM `modulabs_project.data2`
WHERE InvoiceNo LIKE'C%'
LIMIT 100;
```

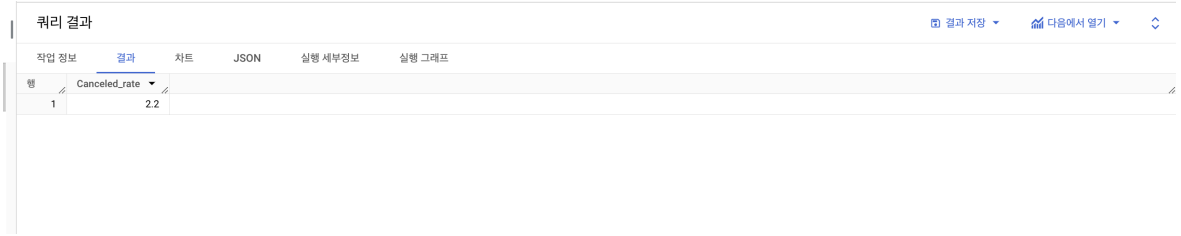
[결과 이미지를 넣어주세요]



- **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 Canceled_rate
FROM `modulabs_project.data2`;
```

[결과 이미지를 넣어주세요]

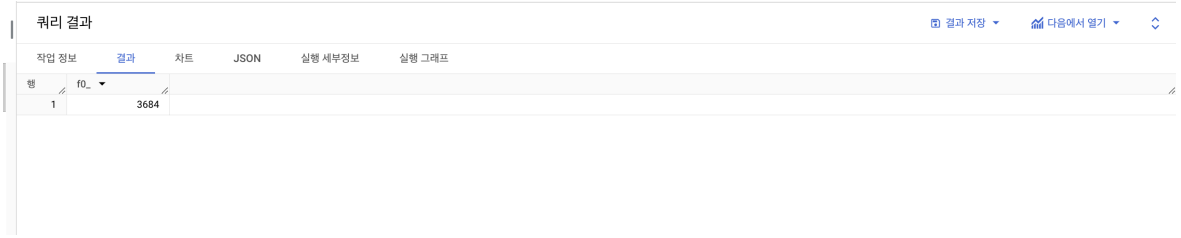


`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
# [[YOUR QUERY]]
SELECT count(DISTINCT stockcode)
FROM `modulabs_project.data2`;
```

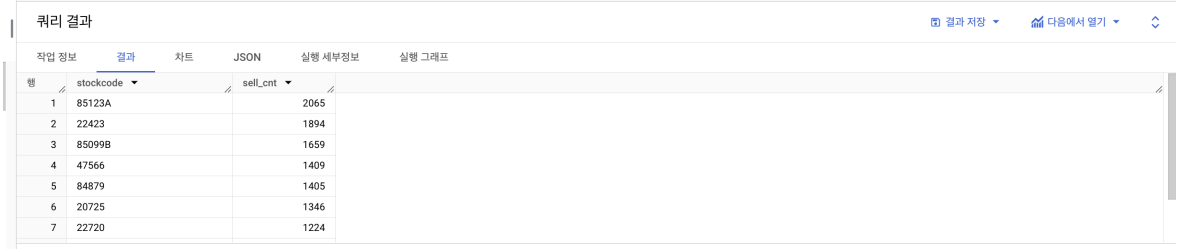
[결과 이미지를 넣어주세요]



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

```
SELECT stockcode, count(*) AS sell_cnt
FROM `modulabs_project.data2`
GROUP BY stockcode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

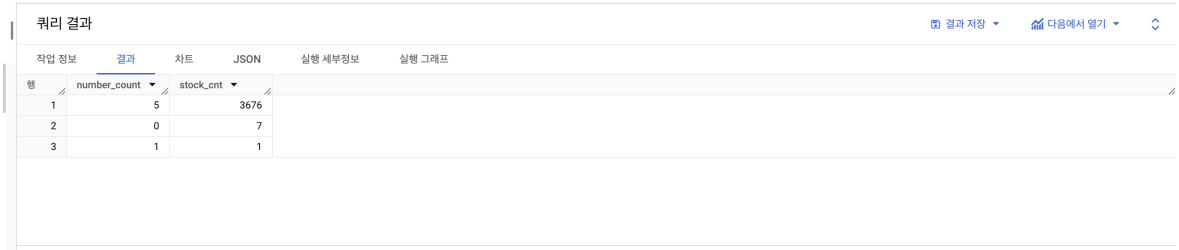


- **`StockCode`의 문자열 내 숫자의 길이를 구해보기**

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM `modulabs_project.data2`
)
SELECT
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
  COUNT(*) AS stock_cnt
FROM UniqueStockCodes
```

```
GROUP BY number_count
ORDER BY stock_cnt DESC;
```

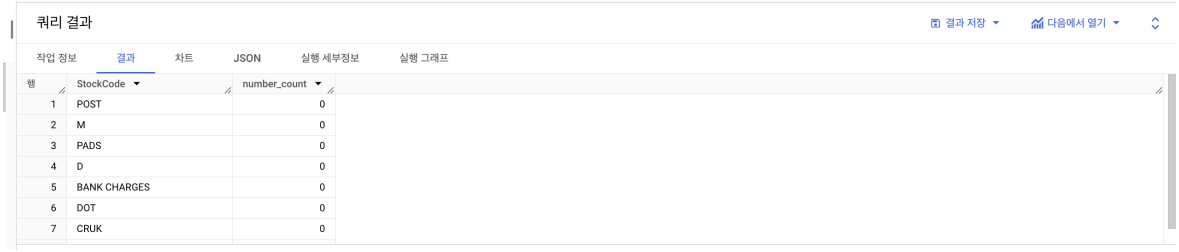
[결과 이미지를 넣어주세요]



- `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 `modulabs_project.data2`
)
WHERE number_count <5;
```

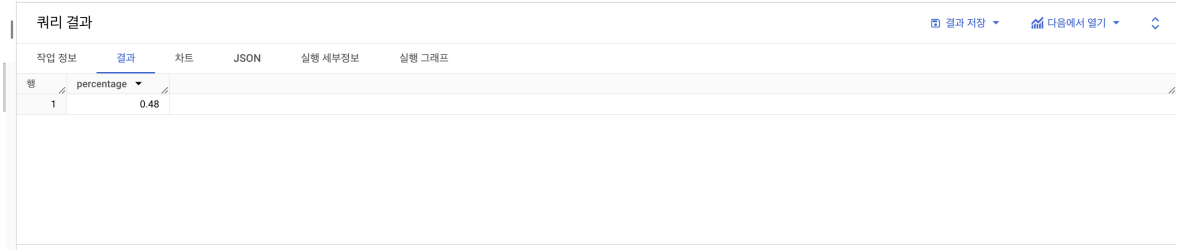
[결과 이미지를 넣어주세요]



- `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)

```
# [[YOUR QUERY]]
WITH canceled_data AS (
    SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `modulabs_project.data2`

)
SELECT
      ROUND(
        (SELECT COUNT(*) FROM canceled_data WHERE number_count < 5)
        /
        (SELECT COUNT(*) FROM `modulabs_project.data2`) * 100,
        2
    ) AS percentage
FROM `modulabs_project.data2`
LIMIT 1;
```

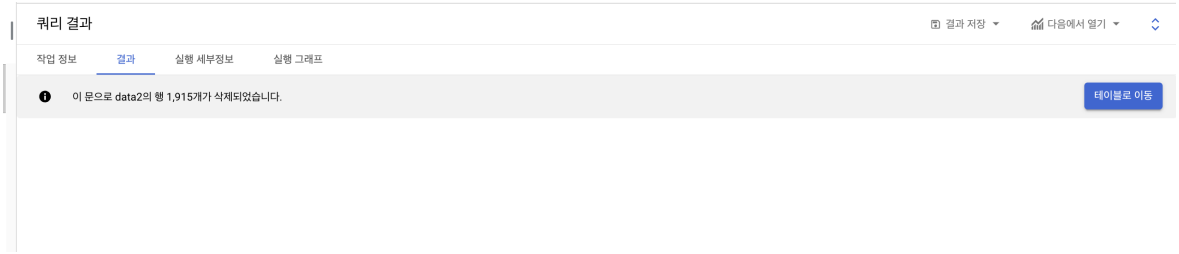
[결과 이미지를 넣어주세요]



- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `modulabs_project.data2`
WHERE StockCode In (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `modulabs_project.data2`
  )
  WHERE number_count <5
  );
```

[결과 이미지를 넣어주세요]

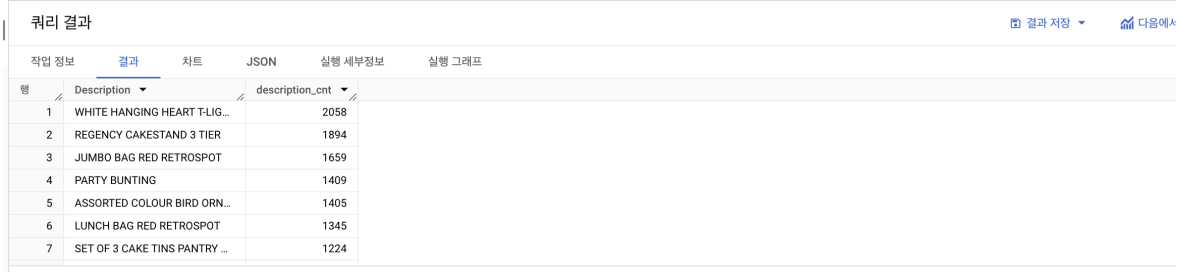


`Description` 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM `modulabs_project.data2`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]



- **대소문자가 혼합된 Description이 있는지 확인하기**

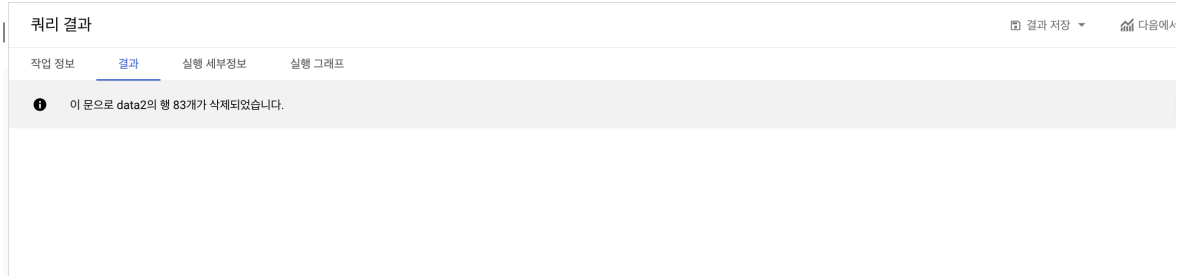
```
SELECT DISTINCT Description
FROM `modulabs_project.data2`
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

[결과 이미지를 넣어주세요]

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `modulabs_project.data2`
WHERE Description IN ('Next Day Carriage','High Resolution Image');
```

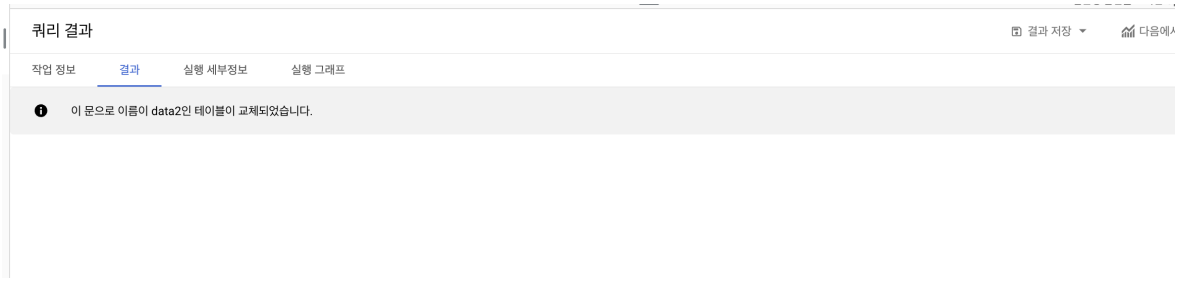
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `modulabs_project.data2` AS
SELECT
  * EXCEPT (Description), -- Description 컬럼 제외
  Description -- 다시 포함 (필터링된 값)
FROM `modulabs_project.data2`
WHERE NOT REGEXP_CONTAINS(Description, r'[a-z]');
```

[결과 이미지를 넣어주세요]

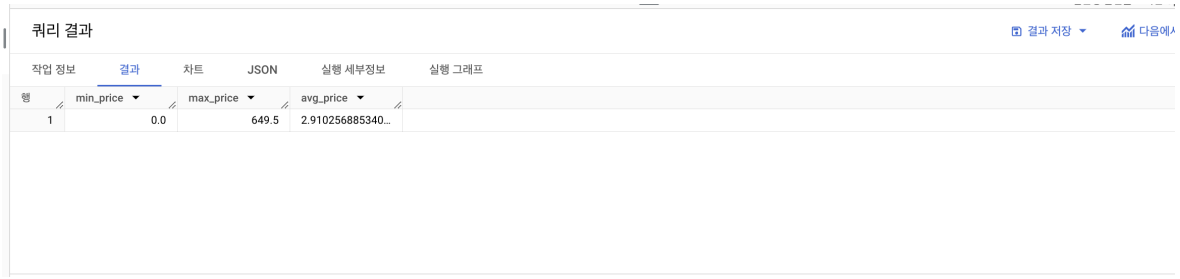


`UnitPrice` 살펴보기

- **`UnitPrice`의 최솟값, 최댓값, 평균을 구하기**

```
CREATE OR REPLACE TABLE `modulabs_project.data2` AS
SELECT
  * EXCEPT (Description), -- Description 컬럼 제외
  Description -- 다시 포함 (필터링된 값)
FROM `modulabs_project.data2`
WHERE NOT REGEXP_CONTAINS(Description, r'[a-z]');
```

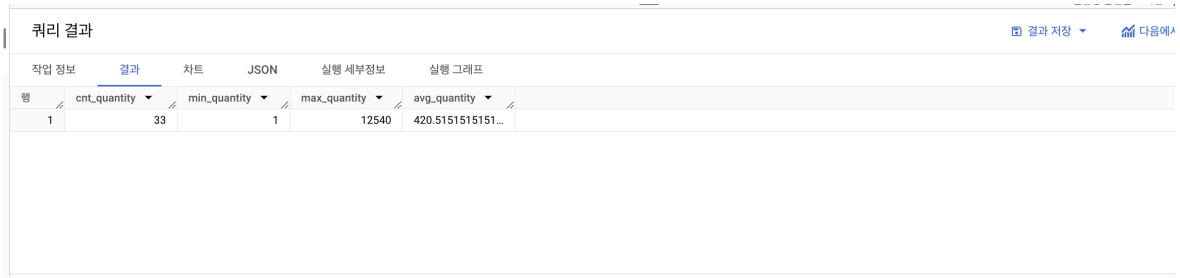
[결과 이미지를 넣어주세요]



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

```
SELECT count(*) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Q
FROM `modulabs_project.data2`
WHERE unitprice=0;
```

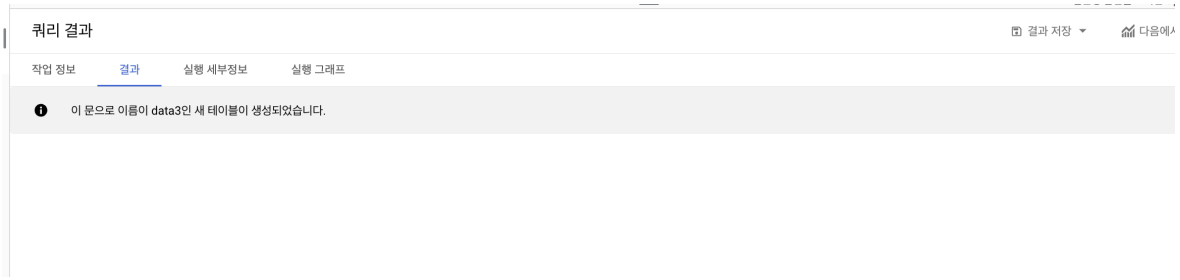
[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE `modulabs_project.data3` AS
SELECT *
FROM `modulabs_project.data2`
WHERE NOT unitprice=0;
```

[결과 이미지를 넣어주세요]



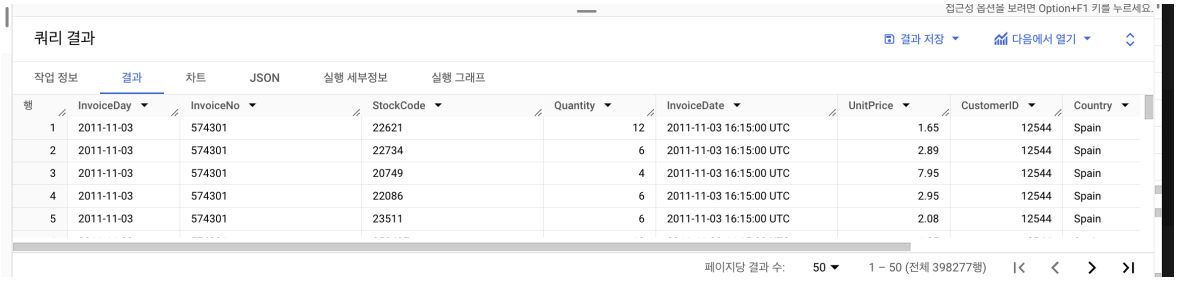
5-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM modulabs_project.data3;
```

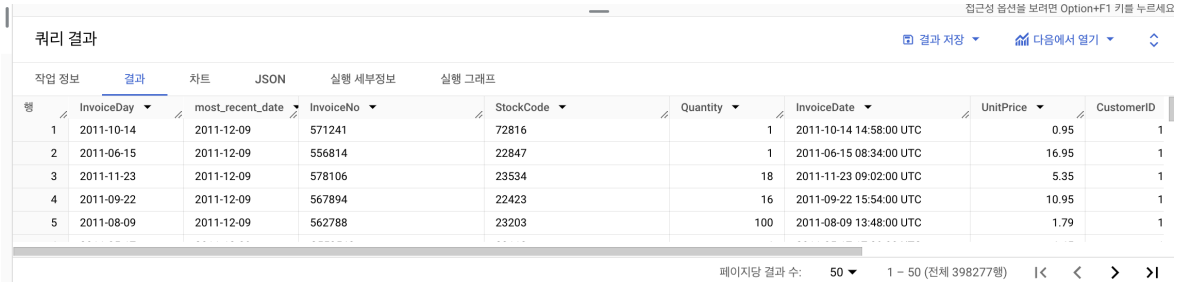
[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  DATE(InvoiceDate) AS InvoiceDay,
  MAX(DATE(InvoiceDate)) OVER() AS most_recent_date,
  *
FROM modulabs_project.data3;
```

[결과 이미지를 넣어주세요]

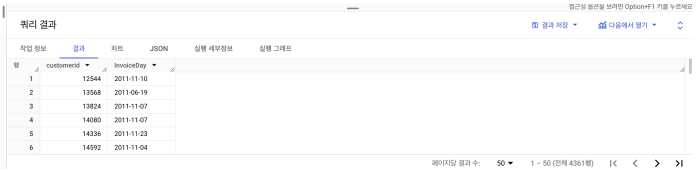


- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

```
SELECT customerid,
  MAX(DATE(InvoiceDate)) AS InvoiceDay,
```

```
FROM modulabs_project.data3
GROUP BY customerid;
```

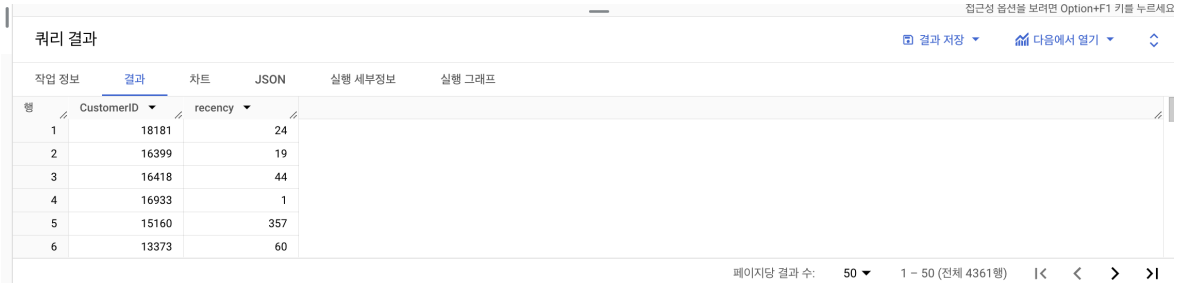
[결과 이미지를 넣어주세요]



- **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 modulabs_project.data3
  GROUP BY CustomerID
);
```

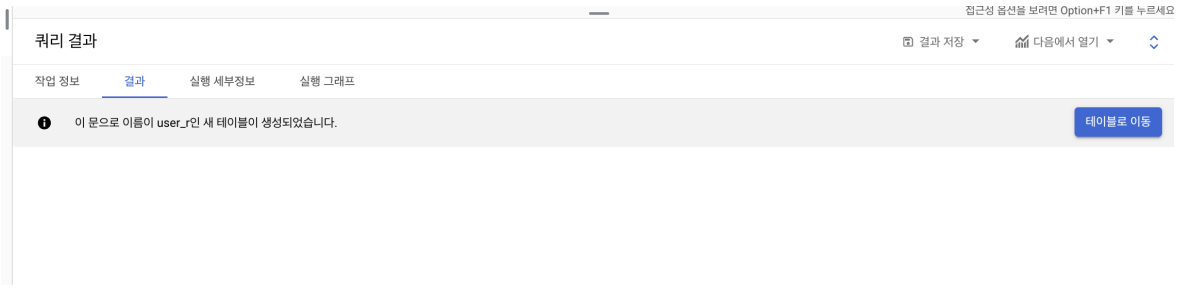
[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE 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 modulabs_project.data3
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

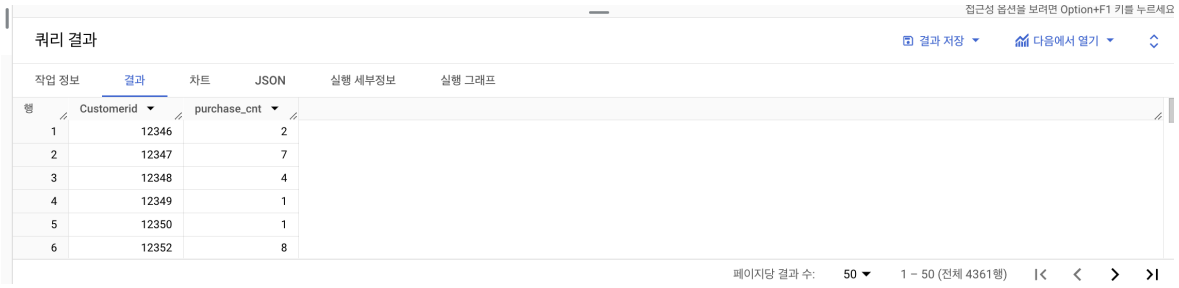


Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT Customerid, count(DISTINCT invoiceNo) AS purchase_cnt
FROM modulabs_project.data3
GROUP BY customerid
ORDER BY customerid;
```

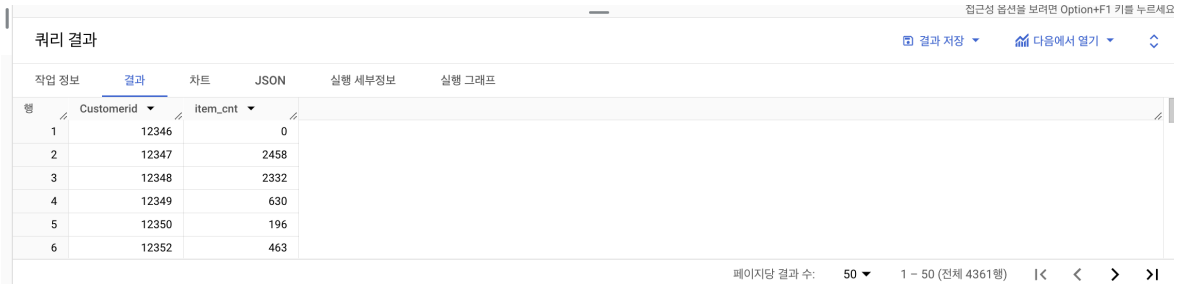
[결과 이미지를 넣어주세요]



- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT Customerid, sum(Quantity) AS item_cnt
FROM modulabs_project.data3
GROUP BY customerid
ORDER BY customerid;
```

[결과 이미지를 넣어주세요]

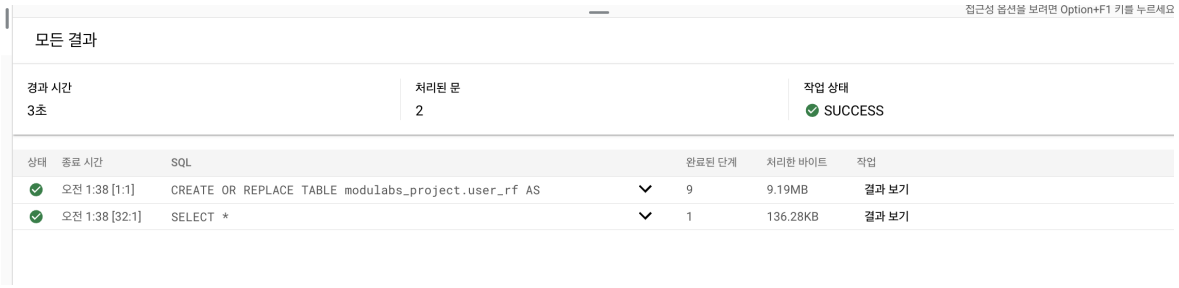


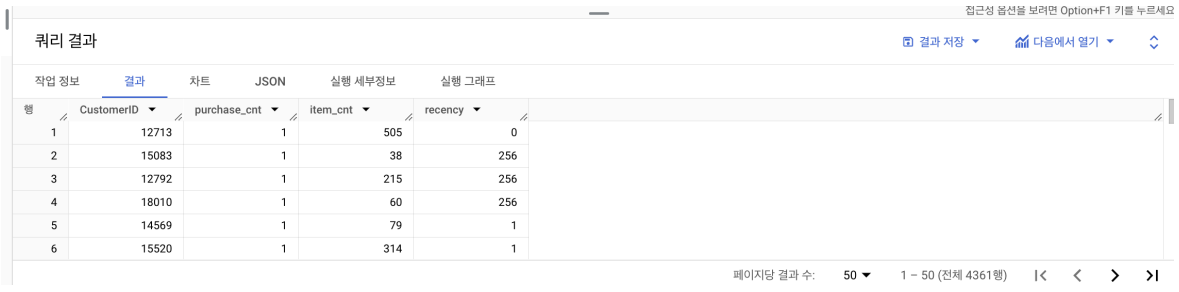
- **전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 `user_rf` 라는 이름의 테이블에 저장하기**

```
CREATE OR REPLACE TABLE modulabs_project.user_rf AS

  -- (1) 전체 거래 건수 계산
```

```
  WITH purchase_cnt AS (
    SELECT Customerid, count(DISTINCT invoiceNo) AS purchase_cnt
    FROM modulabs_project.data3
    GROUP BY customerid
    ORDER BY customerid

  ),

  -- (2) 구매한 아이템 총 수량 계산
  item_cnt AS (
    SELECT Customerid, sum(Quantity) AS item_cnt
    FROM modulabs_project.data3
    GROUP BY customerid
    ORDER 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 modulabs_project.user_r AS ur
    ON pc.CustomerID = ur.CustomerID;

SELECT *
FROM modulabs_project.user_rf;
```

[결과 이미지를 넣어주세요]





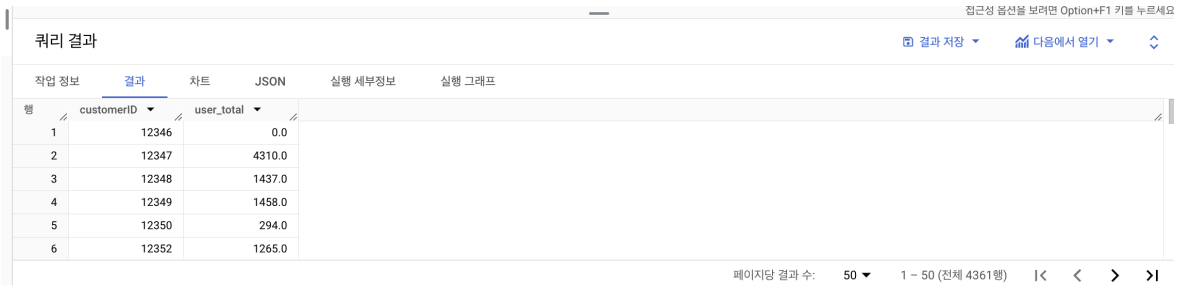
Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT customerID, ROUND(sum(UnitPrice*Quantity)) AS user_total
FROM modulabs_project.data3
```

```
GROUP BY customerID
ORDER BY customerID;
```

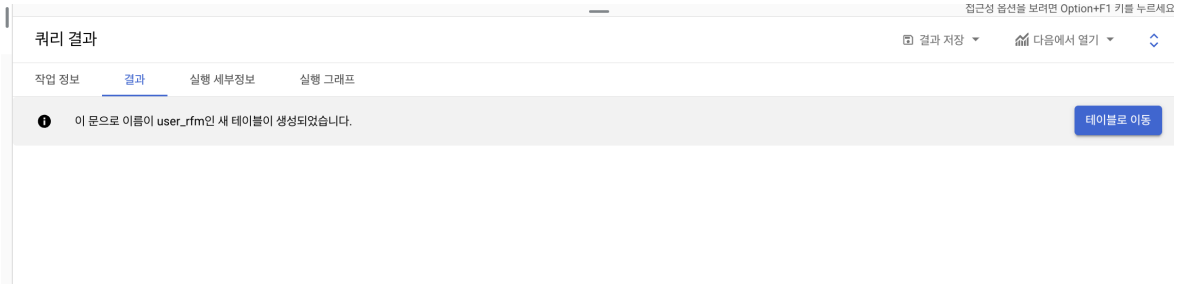
[결과 이미지를 넣어주세요]



- **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 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/ut.count_ctn) AS user_average
FROM modulabs_project.user_rf rf
LEFT JOIN (
 SELECT  customerID, ROUND(sum(UnitPrice*Quantity)) AS user_total,
 count(DISTINCT InvoiceNo) AS count_ctn
FROM modulabs_project.data3
GROUP BY customerID
ORDER BY customerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

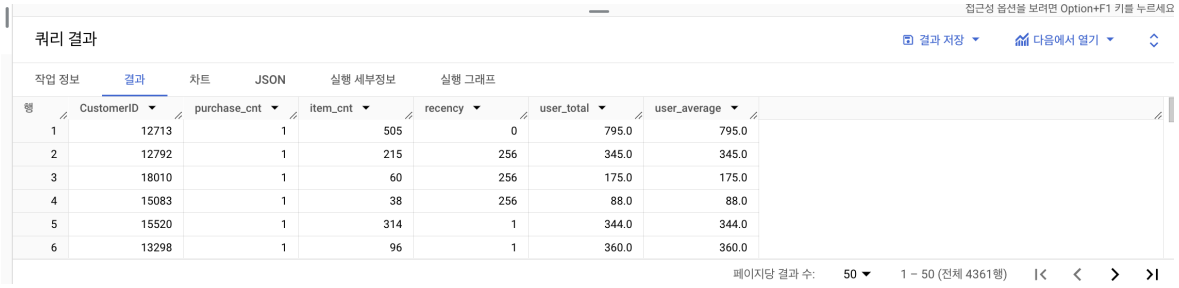


RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
# [[YOUR QUERY]];
SELECT *
FROM `modulabs_project.user_rfm`;
```

[결과 이미지를 넣어주세요]



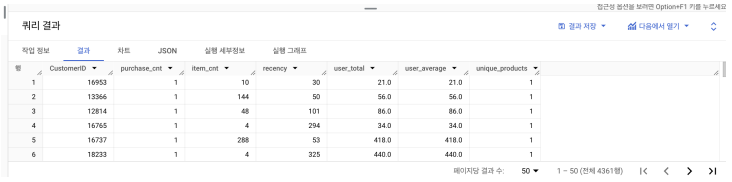
5-8. 추가 Feature 추출

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;

SELECT *
FROM modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]



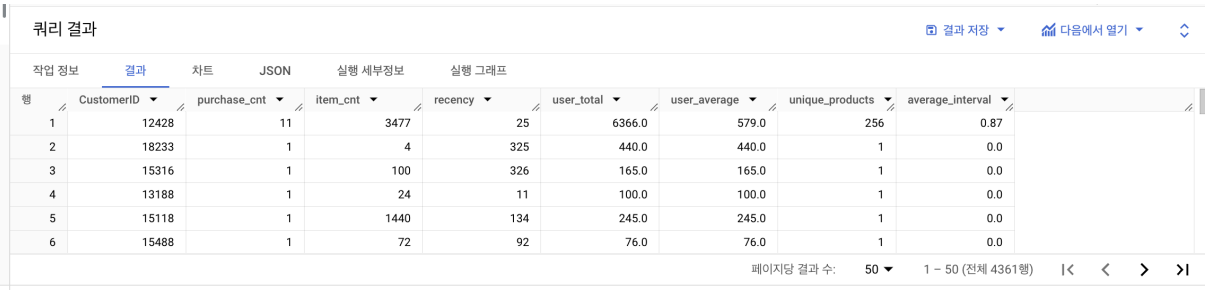
2. 평균 구매 주기

- **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 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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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;

SELECT *
FROM `modulabs_project.user_data`;
```

[결과 이미지를 넣어주세요]



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
1	12428	11	3477	25	6366.0	579.0	256	0.87
2	18233	1	4	325	440.0	440.0	1	0.0
3	15316	1	100	326	165.0	165.0	1	0.0
4	13188	1	24	11	100.0	100.0	1	0.0
5	15118	1	1440	134	245.0	245.0	1	0.0
6	15488	1	72	92	76.0	76.0	1	0.0

페이지당 결과 수: 50 | 1 – 50 (전체 4361행)

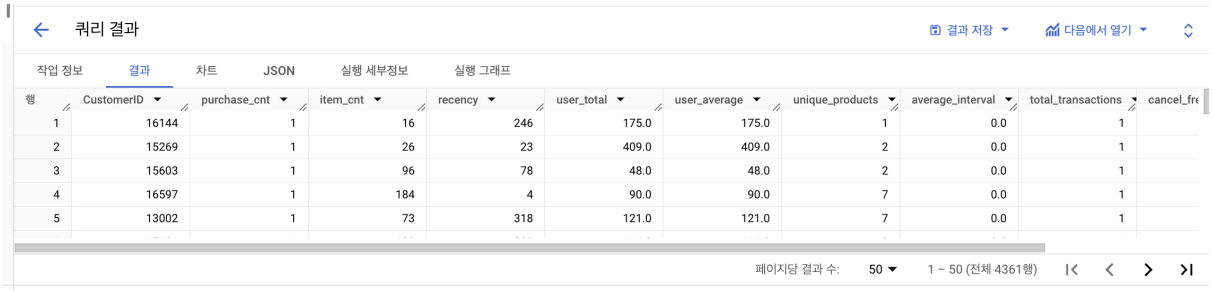
3. 구매 취소 경향성

- **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 modulabs_project.user_data_final 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
    modulabs_project.data3
GROUP BY
    CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), t.cancel_frequency/t.total_transactions AS cancel_rate
FROM `modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.customerid=t.customerid;
```

```
SELECT *
FROM modulabs_project.user_data_final ;
```

[결과 이미지를 넣어주세요]

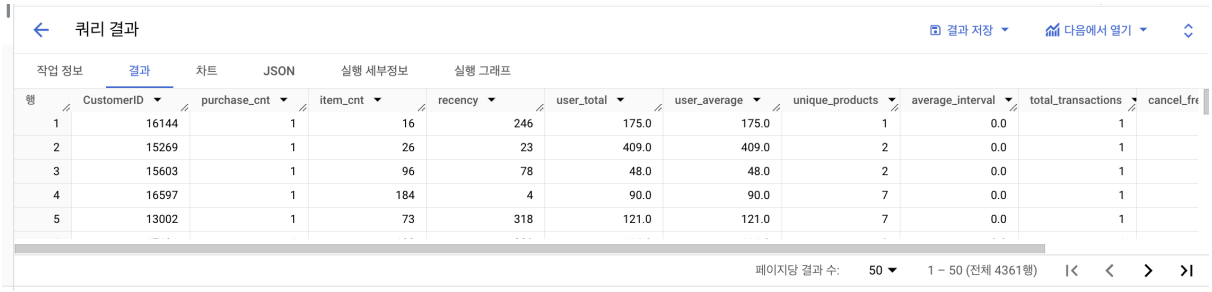


행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_fre
1	16144	1	16	246	175.0	175.0	1	0.0	1	
2	15269	1	26	23	409.0	409.0	2	0.0	1	
3	15603	1	96	78	48.0	48.0	2	0.0	1	
4	16597	1	184	4	90.0	90.0	7	0.0	1	
5	13002	1	73	318	121.0	121.0	7	0.0	1	

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

```
# [[YOUR QUERY]];
SELECT *
FROM modulabs_project.user_data_final ;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_fre
1	16144	1	16	246	175.0	175.0	1	0.0	1	
2	15269	1	26	23	409.0	409.0	2	0.0	1	
3	15603	1	96	78	48.0	48.0	2	0.0	1	
4	16597	1	184	4	90.0	90.0	7	0.0	1	
5	13002	1	73	318	121.0	121.0	7	0.0	1	

마치며

1. 데이터 분석에서 중요한 점

- 프로젝트를 직접 진행해 보기 전까지는 쿼리를 짜는 코딩능력같은 '하드스킬'이 가장 중요하고 생각했는데,
  실제로 프로젝트를 진행해보면서 논리력과 데이터를 해석하는 능력 등의 '소프트 스킬'이 더 중요하다는 것을 알았다.
  - 가이드가 없이 전체 과정을 혼자 진행하게 된다면, 결측치와 이상치를 어떻게 구분할 것이고 어떻게 처리할 것인지 결정하는 것 등의 과정이 어려울 것이라고 예상된다.
  - 왜 채용에서 데이터 관련 자격증보다 포트폴리오를 우선적으로 중요하게 보는지 이해할 수 있었음
- 무작정 분석하기보다, 데이터에 대한 이해를 우선적으로 하는 것이 필요하다.
  - 데이터를 잘 이해하고 있어야 어떻게 다룰 것인지 기획/설계를 할 수 있다.
  - 그런 의미에서 '도메인 지식'이 정말 중요할 것이고, 내가 강점이 되는 도메인이 어딘지 파악하고 포트폴리오를 그쪽으로 쌓는 것이 중요할 것 같다.

2. 앞으로의 학습 방향성

- 익숙한 데이터를 가지고 혼자서 가이드 없이 새로운 방향으로 분석하는 연습을 해봐야겠다.
- 고객 로그데이터로 할 수 있는 것들이 무궁무진하게 많을 같은데, 퍼널 분석/ 코호트 분석도 해보고 싶다.

- SQL을 까먹지 않게 코딩테스트 연습을 꾸준히 진행해야겠고, '정답인 쿼리'는 없지만 '최적의 쿼리'는 있는 것 같으니 항상 좋고 간단한 쿼리를 짜기 위해 고민해야겠다.