고객을 세그먼테이션하자 [프로젝트]

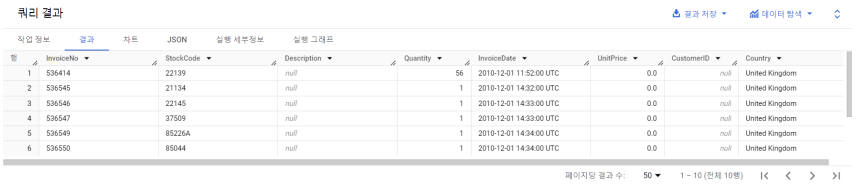
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
select *
from genial-shuttle-439400-e6.modulabs_project.data
limit 10
```

[결과 이미지를 넣어주세요]

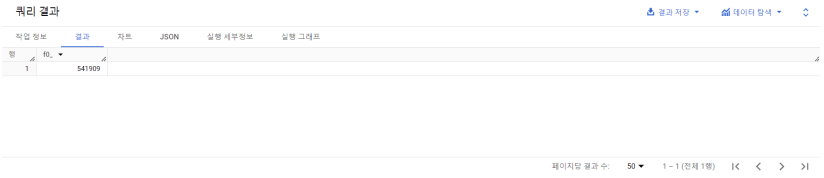


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	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**

```
select count(*)
from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]



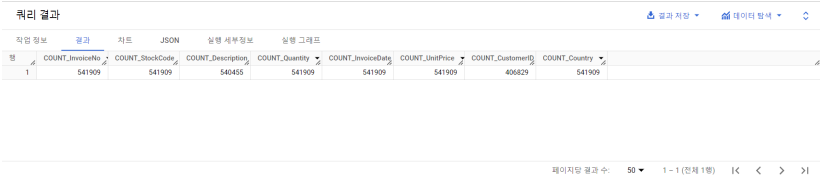
행	f0_
1	541909

데이터 수 세기

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

```
select count(invoiceno) COUNT_InvoiceNo,
      count(stockcode) COUNT_StockCode,
      count(description) COUNT_Description,
      count(quantity) COUNT_Quantity,
      count(invoicedate) COUNT_InvoiceDate,
      count(unitprice) COUNT_UnitPrice,
      count(customerid) COUNT_CustomerID,
      count(country) COUNT_Country
from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]



행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country
1	541909	541909	540455	541909	541909	541909	406829	541909

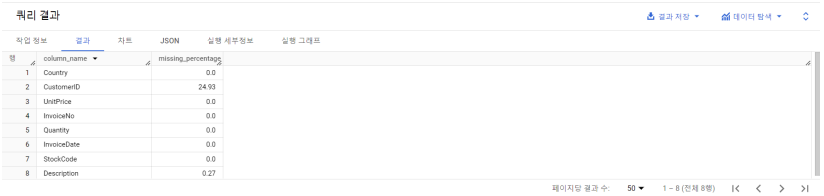
11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

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

```
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM genial-shuttle-439400-e6.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_percent
FROM genial-shuttle-439400-e6.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_perce
FROM genial-shuttle-439400-e6.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_percenta
FROM genial-shuttle-439400-e6.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_perce
FROM genial-shuttle-439400-e6.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_percent
FROM genial-shuttle-439400-e6.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_percen
FROM genial-shuttle-439400-e6.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_percentag
FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

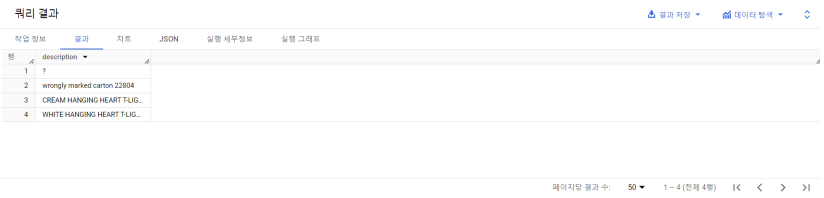


결측치 처리 전략

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

```
select distinct description
from genial-shuttle-439400-e6.modulabs_project.data
where StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

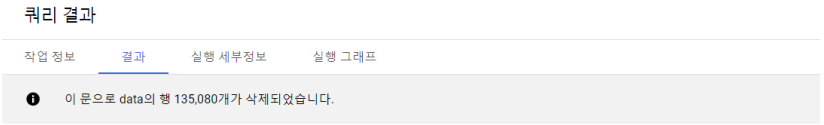


결측치 처리

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

```
DELETE FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE Description is null or customerid is null;
```

[결과 이미지를 넣어주세요]



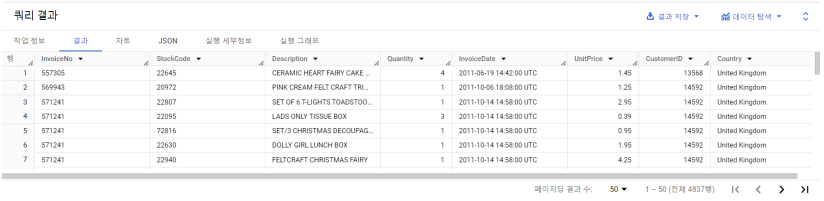
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
select *
from genial-shuttle-439400-e6.modulabs_project.data
group by 1, 2, 3, 4, 5, 6, 7, 8
having count(*) > 1
```

[결과 이미지를 넣어주세요]



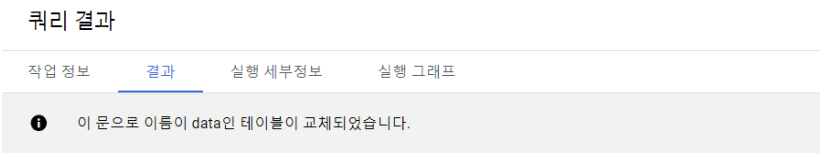
count가 1보다 큰 데이터 개수 : 전체 행 개수

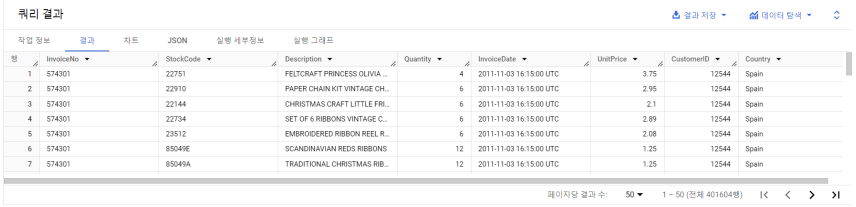
중복값 처리

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

```
create or replace table genial-shuttle-439400-e6.modulabs_project.data as
select distinct *
            from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]





11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

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

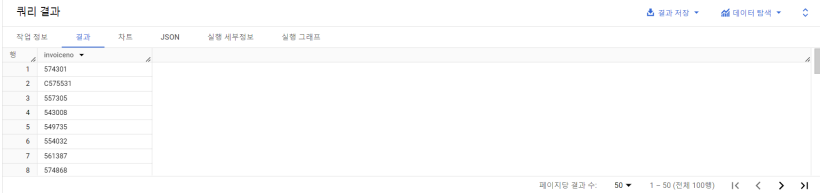
```
select count(distinct invoiceno)
from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

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

```
select distinct invoiceno from genial-shuttle-439400-e6.modulabs_project.data
limit 100
```

[결과 이미지를 넣어주세요]

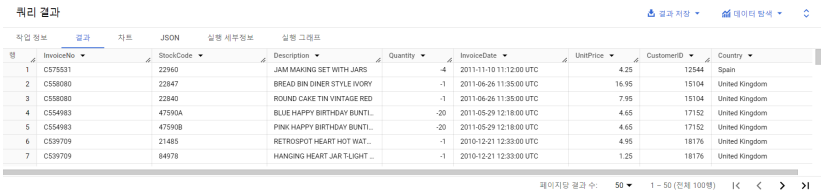


- **`InvoiceNo`가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)**

```
SELECT *
FROM genial-shuttle-439400-e6.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(invoiceno) * 100, 1)
FROM genial-shuttle-439400-e6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

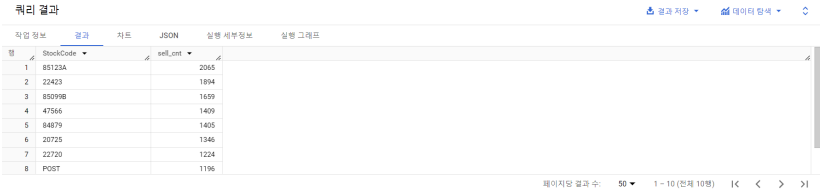
```
select count(distinct stockcode)
from genial-shuttle-439400-e6.modulabs_project.data
```

[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 genial-shuttle-439400-e6.modulabs_project.data
group by stockcode
ORDER BY sell_cnt DESC
limit 10;
```

[결과 이미지를 넣어주세요]



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

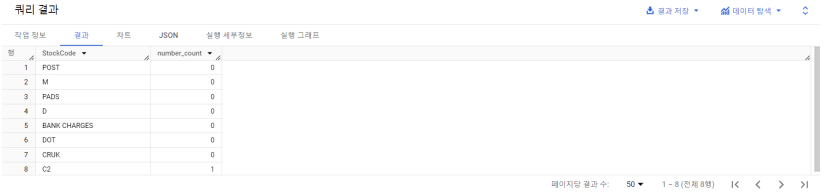
행	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196

페이지당 결과 수: 50 1 – 10 (전체 10행)

- `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 genial-shuttle-439400-e6.modulabs_project.data
)
WHERE number_count < 5;
```

[결과 이미지를 넣어주세요]



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

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

페이지당 결과 수: 50 1 – 8 (전체 8행)

- `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 stockcode in (SELECT DISTINCT StockCode
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM genial-shuttle-439400-e6.modulabs_project.data
)
WHERE number_count < 5)
 THEN 1 ELSE 0 END)/count(stockcode) * 100, 2)
  FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

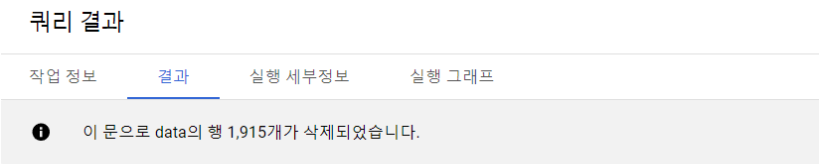
행	f0_
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.data
    )
    WHERE number_count < 5
)
```

[결과 이미지를 넣어주세요]

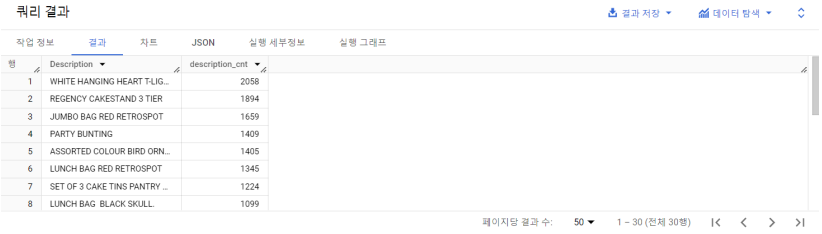


`Description` 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM genial-shuttle-439400-e6.modulabs_project.data
group by description
order by description_cnt desc
limit 30;
```

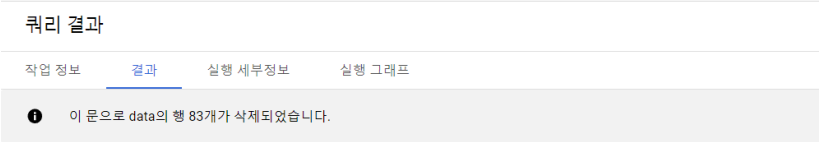
[결과 이미지를 넣어주세요]



- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE
description = 'Next Day Carriage'
or description = 'High Resolution Image'
```

[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE genial-shuttle-439400-e6.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  upper(Description) as Description
FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다.

`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 genial-shuttle-439400-e6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

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

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

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

```
SELECT count(unitprice) AS cnt_quantity, min(quantity) AS min_quantity, max(quantity) AS max_quantit
FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE unitprice = 0;
```

[결과 이미지를 넣어주세요]

쿼리 결과

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

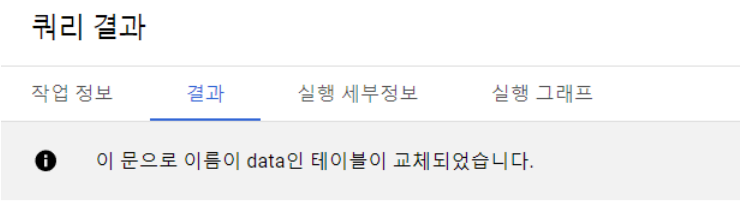
행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

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

```
CREATE OR REPLACE TABLE genial-shuttle-439400-e6.modulabs_project.data AS
SELECT *
```

```
FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE unitprice != 0;
```

[결과 이미지를 넣어주세요]



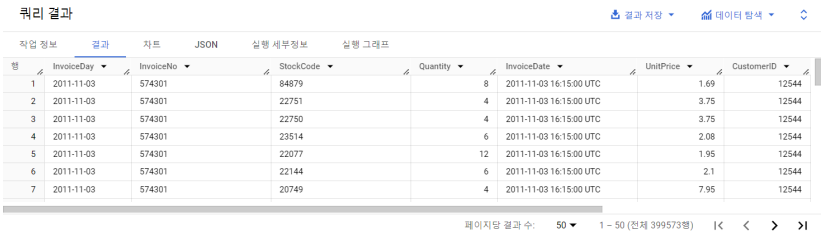
11-7. RFM 스코어

Recency

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

```
SELECT date(invoicedate) AS InvoiceDay, *
FROM genial-shuttle-439400-e6.modulabs_project.data
```

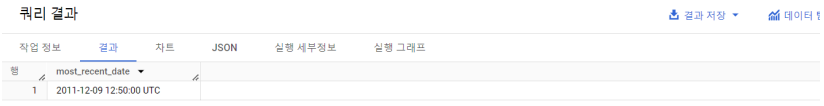
[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT max(invoicedate) AS most_recent_date
FROM genial-shuttle-439400-e6.modulabs_project.data
```

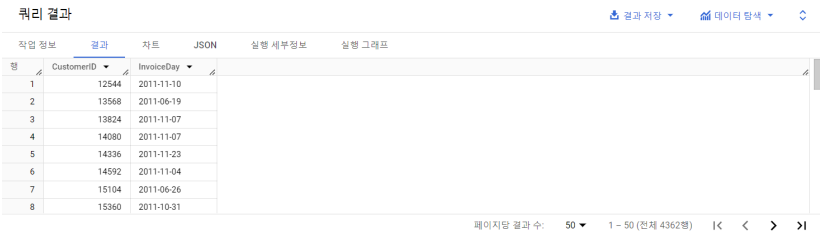
[결과 이미지를 넣어주세요]



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

```
SELECT
    CustomerID,
    max(date(invoicedate)) AS InvoiceDay
FROM genial-shuttle-439400-e6.modulabs_project.data
group by customerID;
```

[결과 이미지를 넣어주세요]

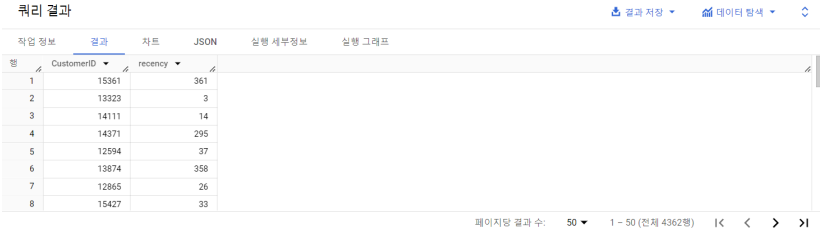


행	CustomerID	InvoiceDay
1	12544	2011-11-10
2	13568	2011-06-19
3	13824	2011-11-07
4	14080	2011-11-07
5	14336	2011-11-23
6	14592	2011-11-04
7	15104	2011-06-26
8	15360	2011-10-31

- **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]

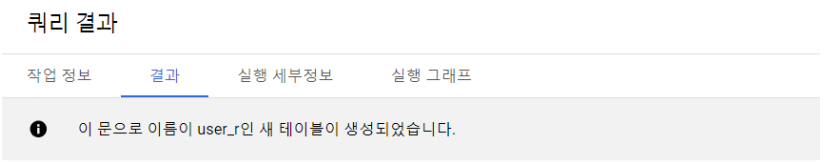


행	CustomerID	recency
1	15361	361
2	13323	3
3	14111	14
4	14371	295
5	12594	37
6	13874	358
7	12865	26
8	15427	33

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

```
CREATE OR REPLACE TABLE genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	recency
1	16446	0
2	13777	0
3	15311	0
4	15910	0
5	17389	0
6	12433	0
7	17581	0
8	12518	0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
select
    customerid,
    count(distinct invoiceno) as purchase_cnt
from genial-shuttle-439400-e6.modulabs_project.data
group by customerid
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	customerid	purchase_cnt
1	12544	2
2	13568	1
3	13824	5
4	14080	1
5	14336	4
6	14592	3
7	15104	3
8	15360	1

페이지당 결과 수: 50 1 – 50 (전체 4362행)

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  sum(quantity) AS item_cnt
FROM genial-shuttle-439400-e6.modulabs_project.data
group by customerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	item_cnt
1	12544	130
2	13568	66
3	13824	768
4	14080	48
5	14336	1759
6	14592	407
7	15104	633
8	15360	223

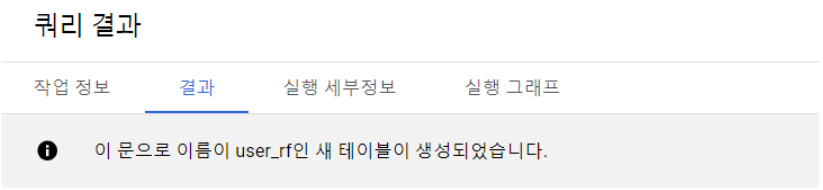
페이지당 결과 수: 50 1 – 50 (전체 4362행)

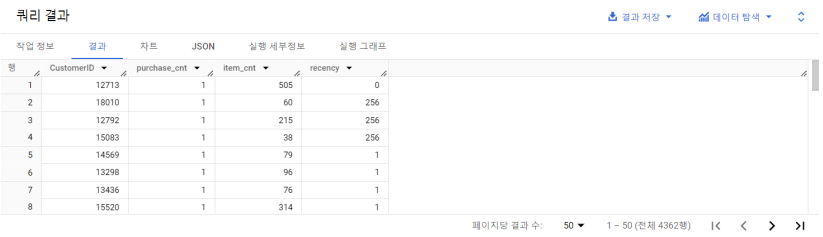
- **전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 `user_rf` 라는 이름의 테이블에 저장하기**

```
CREATE OR REPLACE TABLE genial-shuttle-439400-e6.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  select
  customerid,
  count(distinct invoiceno) as purchase_cnt
  from genial-shuttle-439400-e6.modulabs_project.data
  group by customerid
),
```

```
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
  CustomerID,
  sum(quantity) AS item_cnt
  FROM genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]



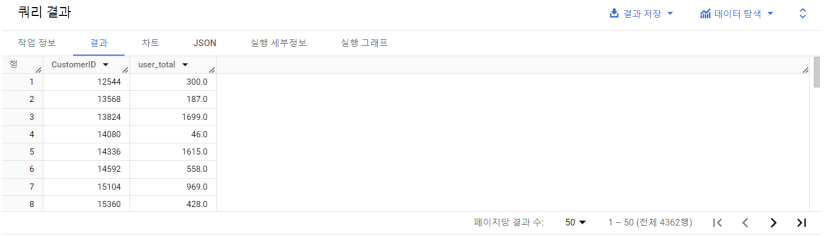


Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  round(sum(unitprice * quantity), 0) AS user_total
FROM genial-shuttle-439400-e6.modulabs_project.data
group by customerid
```

[결과 이미지를 넣어주세요]



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

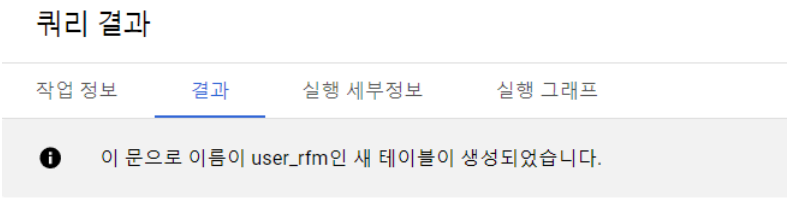
행	CustomerID	user_total
1	12544	300.0
2	13568	187.0
3	13824	1699.0
4	14080	46.0
5	14336	1615.0
6	14592	558.0
7	15104	969.0
8	15360	428.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

- **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 genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    round(sum(unitprice * quantity), 0) AS user_total
  FROM genial-shuttle-439400-e6.modulabs_project.data
  group by customerid
) ut
ON rf.CustomerID = ut.CustomerID;
```

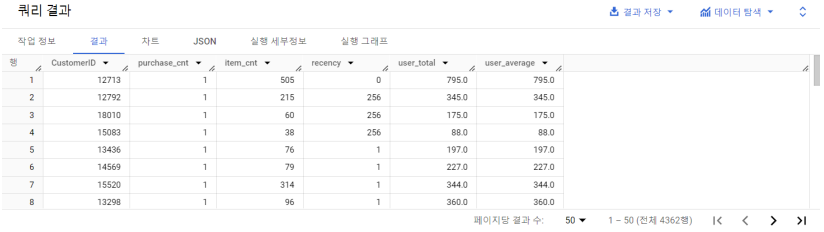
[결과 이미지를 넣어주세요]



쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	12792	1	215	256	345.0	345.0
3	18010	1	60	256	175.0	175.0
4	15083	1	38	256	88.0	88.0
5	13436	1	76	1	197.0	197.0
6	14569	1	79	1	227.0	227.0
7	15520	1	314	1	344.0	344.0
8	13298	1	96	1	360.0	360.0

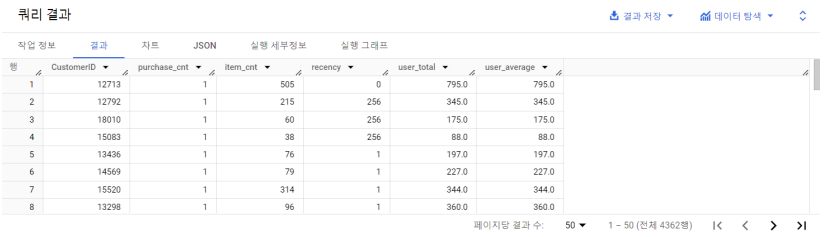
페이지당 결과 수: 50 1 – 50 (전체 4362행)

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select * from genial-shuttle-439400-e6.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	12792	1	215	256	345.0	345.0
3	18010	1	60	256	175.0	175.0
4	15083	1	38	256	88.0	88.0
5	13436	1	76	1	197.0	197.0
6	14569	1	79	1	227.0	227.0
7	15520	1	314	1	344.0	344.0
8	13298	1	96	1	360.0	360.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

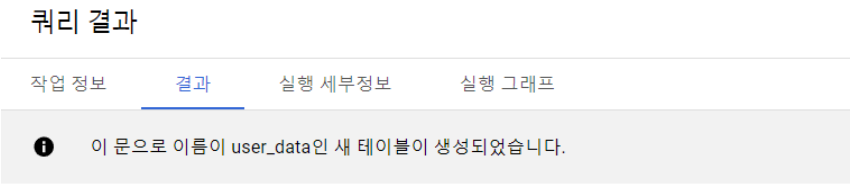
11-8. 추가 Feature 추출

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;
```

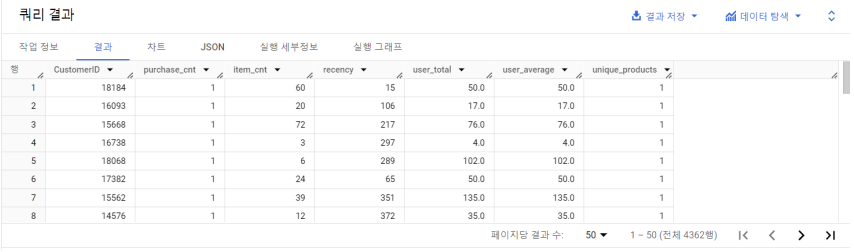
[결과 이미지를 넣어주세요]



쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	18184	1	60	15	50.0	50.0	1
2	16093	1	20	106	17.0	17.0	1
3	15668	1	72	217	76.0	76.0	1
4	16738	1	3	297	4.0	4.0	1
5	18068	1	6	289	102.0	102.0	1
6	17382	1	24	65	50.0	50.0	1
7	15562	1	39	351	135.0	135.0	1
8	14576	1	12	372	35.0	35.0	1

페이지당 결과 수: 50 1 – 50 (전체 4362행)

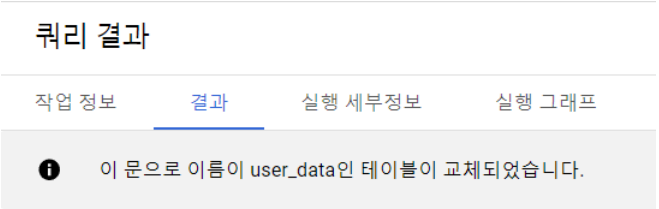
2. 평균 구매 주기

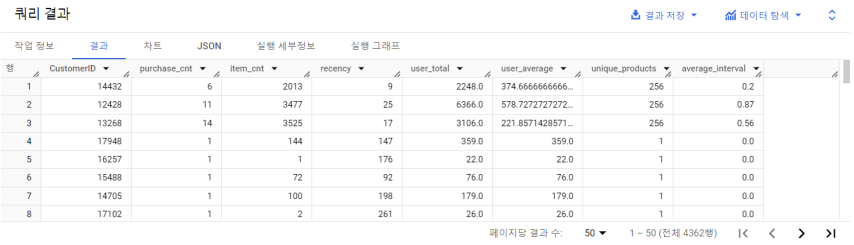
- **고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)**

- **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_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;
```

[결과 이미지를 넣어주세요]





행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	14432	6	2013	9	2248.0	374.6666666666...	256	0.2
2	12428	11	3477	25	6366.0	578.7272727272...	256	0.87
3	13268	14	3525	17	3106.0	221.8571428571...	256	0.56
4	17948	1	144	147	359.0	359.0	1	0.0
5	16257	1	1	176	22.0	22.0	1	0.0
6	15488	1	72	92	76.0	76.0	1	0.0
7	14705	1	100	198	179.0	179.0	1	0.0
8	17102	1	2	261	26.0	26.0	1	0.0

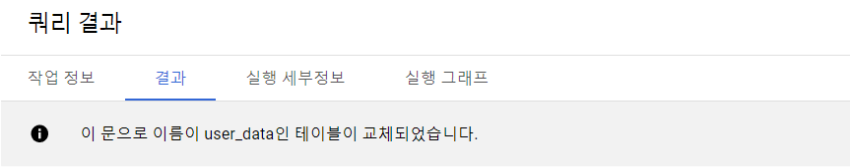
3. 구매 취소 경향성

- **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 genial-shuttle-439400-e6.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    count(customerid) AS total_transactions,
    count(case when invoiceno like 'C%' then 1 else 0 end) AS cancel_frequency
```

```
  FROM genial-shuttle-439400-e6.modulabs_project.data
  group by customerid
)

SELECT u.*, t.* EXCEPT(CustomerID), round(t.cancel_frequency / t.total_transactions,2) AS cancel_rate
FROM genial-shuttle-439400-e6.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.customerid = t.customerid;
```

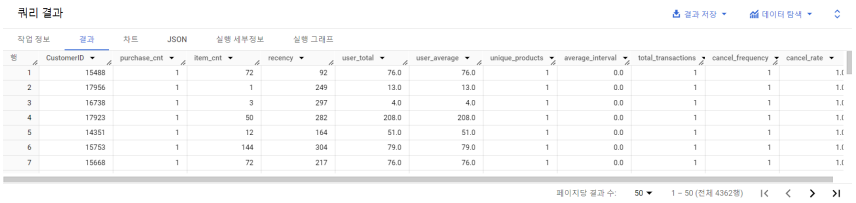
[결과 이미지를 넣어주세요]



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

```
select * from genial-shuttle-439400-e6.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

sql에 대해서는 어느 정도 공부했었지만 sql의 활용에 대해서는 문제 풀이 말고는 겪어본 적이 없었다.

그래서 오늘 진행한 세그먼테이션 프로젝트가 더 의미 있을 수 있었다.

어제 배웠던 퍼널 분석, 리텐션 분석, rfm 분석 모두 새로 배우는 거라 흥미로웠고,

실제 현업에서 많이 활용되는 rfm 분석에 대해 (실질적으로는 맛보기라고 봐야겠지만) sql 측면에서는 데이터를 정제하고 가공하는 일련의 과정을 경험해 볼 수 있어 좋았다.

사실 데이터사이언스를 공부하면서 어떤 식으로 프로젝트를 진행해야 할지, 어떤 문제를 어떻게 해결할지에 대해서 막막한 부분들이 많았는데,

문제가 주어졌을 때 어떤 식으로 파악하고 문제를 해결할 수 있다는 가이드라인이 어느 정도 머리에 각인되었다.

물론 새로운 문제가 주어졌을 때 척척 풀어낼 수 있다고 보장하지는 못하겠지만

새로운 데이터 분석 과제를 몇 번 더 직접 수행해보면 익숙하게 만들 수 있을 것 같다.