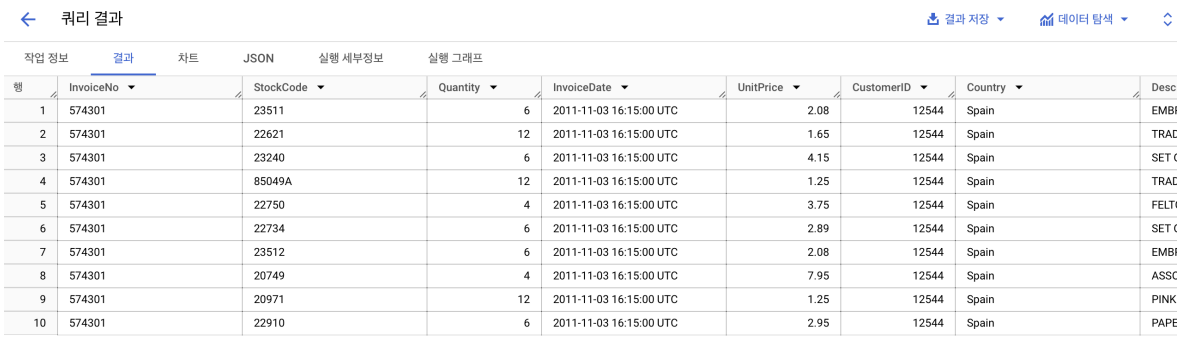
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT  *
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
LIMIT 10;
```



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Desc
1	574301	23511	6	2011-11-03 16:15:00 UTC	2.08	12544	Spain	EMBI
2	574301	22621	12	2011-11-03 16:15:00 UTC	1.65	12544	Spain	TRAD
3	574301	23240	6	2011-11-03 16:15:00 UTC	4.15	12544	Spain	SET (
4	574301	85049A	12	2011-11-03 16:15:00 UTC	1.25	12544	Spain	TRAD
5	574301	22750	4	2011-11-03 16:15:00 UTC	3.75	12544	Spain	FELT
6	574301	22734	6	2011-11-03 16:15:00 UTC	2.89	12544	Spain	SET (
7	574301	23512	6	2011-11-03 16:15:00 UTC	2.08	12544	Spain	EMBI
8	574301	20749	4	2011-11-03 16:15:00 UTC	7.95	12544	Spain	ASSO
9	574301	20971	12	2011-11-03 16:15:00 UTC	1.25	12544	Spain	PINK
10	574301	22910	6	2011-11-03 16:15:00 UTC	2.95	12544	Spain	PAPE

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	f0_
1	541909

데이터 수 세기

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

```
SELECT
    COUNT(InvoiceNo) AS InvoiceNo_count,
    COUNT(StockCode) AS StockCode_count,
    COUNT(Description) AS Description_count,
    COUNT(Quantity) AS Quantity_count,
    COUNT(InvoiceDate) AS InvoiceDate_count,
    COUNT(UnitPrice) AS UnitPrice_count,
    COUNT(CustomerID) AS CustomerID_count,
```

```
    COUNT(Country) AS Country_count
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```

행	InvoiceNo_count	StockCode_count	Description_count	Quantity_count	InvoiceDate_count	UnitPrice_count	CustomerID_count	Country_count
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_perc
FROM `axiomatic-jet-425505-j5.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_perc
FROM `axiomatic-jet-425505-j5.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_pe
FROM `axiomatic-jet-425505-j5.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_perce
FROM `axiomatic-jet-425505-j5.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_pe
FROM `axiomatic-jet-425505-j5.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_perc
FROM `axiomatic-jet-425505-j5.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_per
FROM `axiomatic-jet-425505-j5.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_percen
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```

← 쿼리 결과

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

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
WHERE StockCode = '85123A';
```

결측치 처리

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

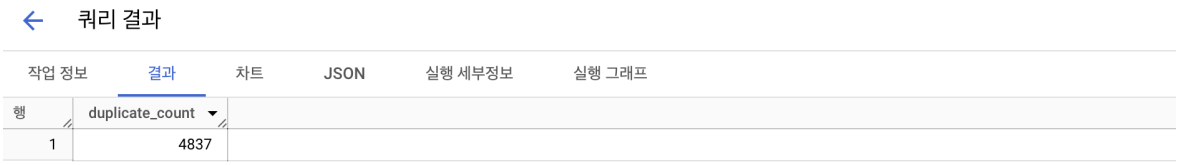
```
DELETE FROM `axiomatic-jet-425505-j5.modulabs_project.data1`
WHERE InvoiceNo IS NULL
   OR StockCode IS NULL
   OR Description IS NULL
   OR Quantity IS NULL
   OR InvoiceDate IS NULL
   OR UnitPrice IS NULL
   OR CustomerID IS NULL
   OR Country IS NULL;
```

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

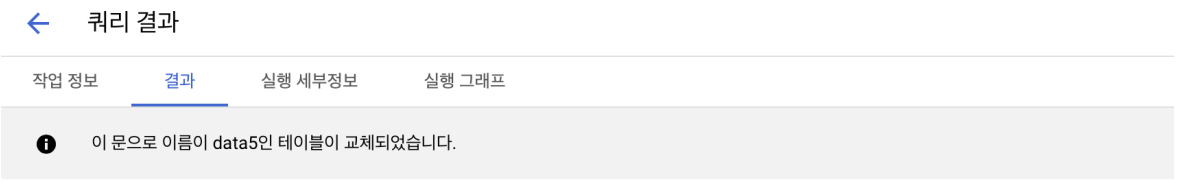
```
SELECT COUNT(*) AS duplicate_count
FROM (
  SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  FROM `axiomatic-jet-425505-j5.modulabs_project.data`
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
) AS subquery;
```



중복값 처리

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

```
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.data5` AS
SELECT DISTINCT *
FROM `axiomatic-jet-425505-j5.modulabs_project.data5`;
```



11-6. 데이터 전처리(3): 오류값 처리

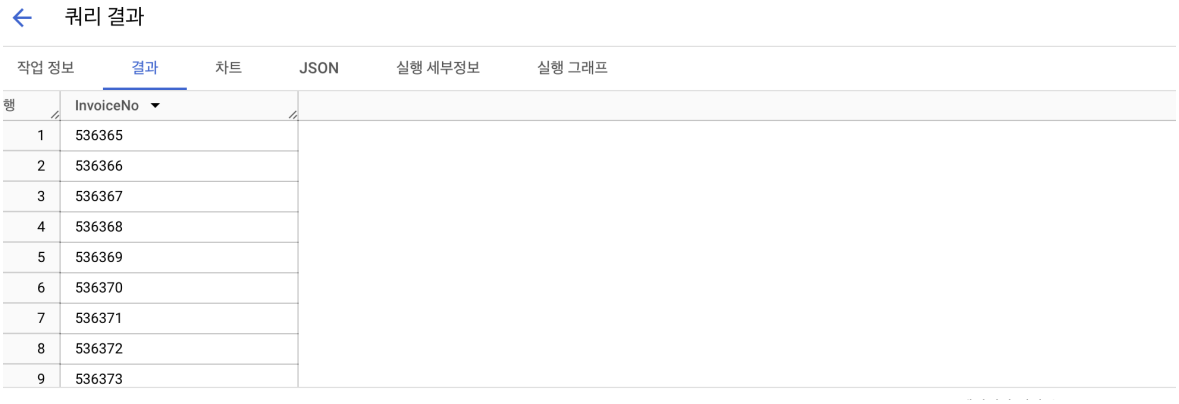
`InvoiceNo` 살펴보기

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

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```

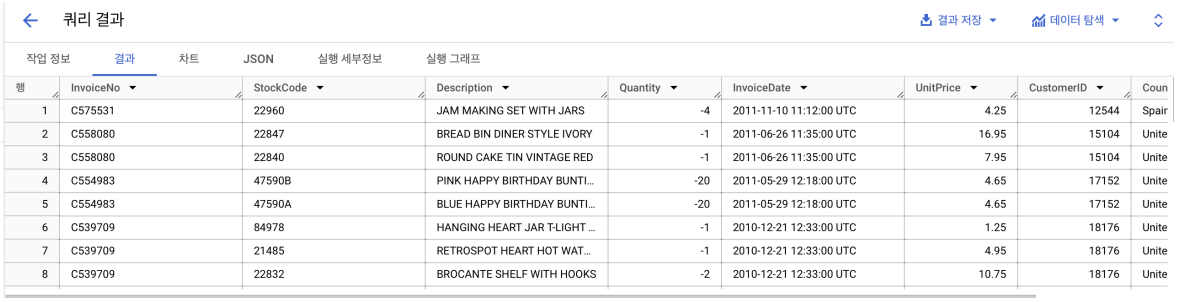
- **고유한 `InvoiceNo`를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
ORDER BY InvoiceNo
LIMIT 100;
```



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

```
SELECT *
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```



- **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 cancelled
```

```
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```

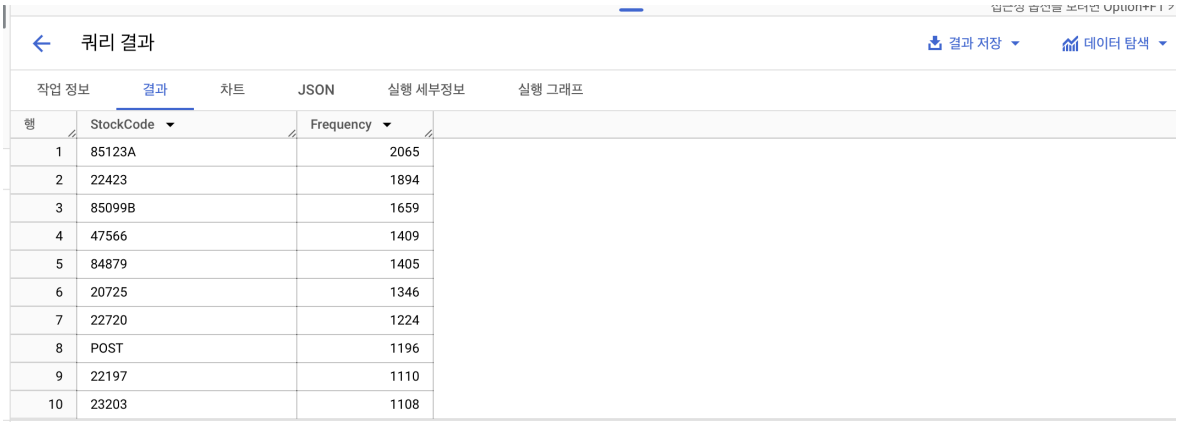
`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `axiomatic-jet-425505-j5.modulabs_project.data1`;
```

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

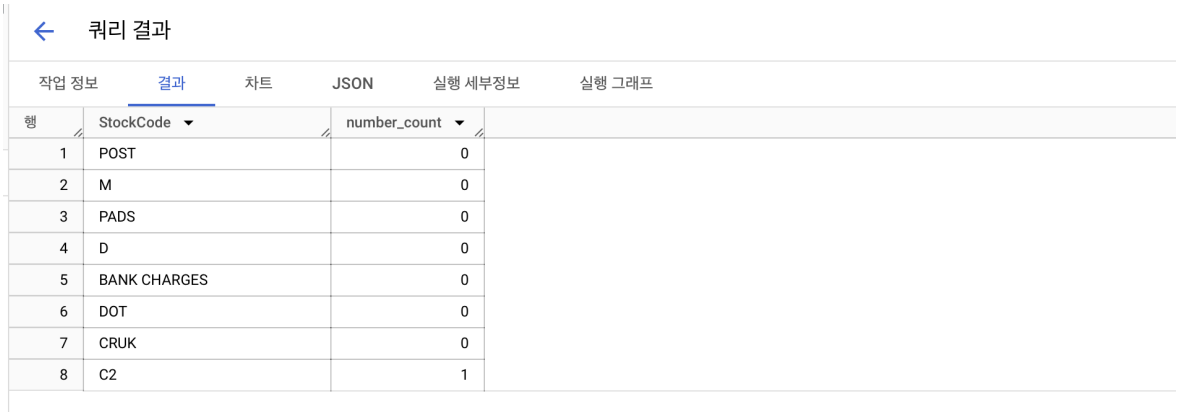
```
SELECT StockCode, COUNT(*) AS Frequency
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
GROUP BY StockCode
ORDER BY Frequency DESC
LIMIT 10;
```



- **`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 `axiomatic-jet-425505-j5.modulabs_project.data1`
)
WHERE number_count IN (0, 1);
```



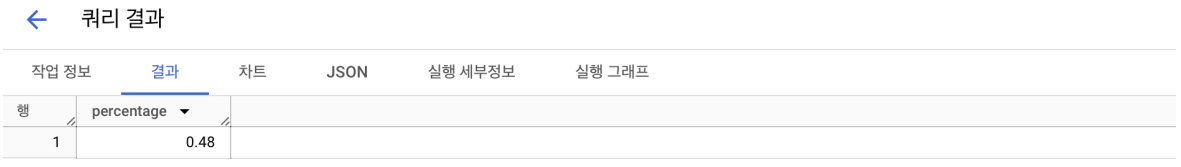
쿼리 결과

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

- **`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 DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
  FROM `axiomatic-jet-425505-j5.modulabs_project.data1`
)
WHERE number_count IN (0, 1);

WITH FilteredData AS (
  SELECT StockCode
  FROM (
    SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
    FROM `axiomatic-jet-425505-j5.modulabs_project.data`
  )
  WHERE number_count IN (0, 1)
)
SELECT
  ROUND((COUNT(*) / (SELECT COUNT(*) FROM `axiomatic-jet-425505-j5.modulabs_project.data`)) * 100, 2
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
WHERE StockCode IN (SELECT StockCode FROM FilteredData);
```



쿼리 결과

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

행	percentage
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
-- 숫자가 0~1인 값을 가진 StockCode를 추출
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.filtered_stockcodes` AS
SELECT StockCode
```

```
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
  FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
)
WHERE number_count IN (0, 1);

-- 해당 StockCode를 포함한 기록을 제외하고 데이터를 다시 테이블에 저장
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.data7_filtered` AS
SELECT *
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
WHERE StockCode NOT IN (SELECT StockCode FROM `axiomatic-jet-425505-j5.modulabs_project.filtered_sto


-- 전체 행 개수
SELECT COUNT(*) AS total_count_before
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`;

-- 숫자가 0~1인 값을 가진 StockCode를 추출
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.filtered_stockcodes` AS
SELECT StockCode
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
  FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
)
WHERE number_count IN (0, 1);

-- 필터링된 데이터로 새로운 테이블 생성
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.data7_filtered` AS
SELECT *
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
WHERE StockCode NOT IN (SELECT StockCode FROM `axiomatic-jet-425505-j5.modulabs_project.filtered_sto

-- 필터링 후의 전체 행 개수
SELECT COUNT(*) AS total_count_after
FROM `axiomatic-jet-425505-j5.modulabs_project.data7_filtered`;

-- 삭제된 행 개수 계산
SELECT
  (SELECT COUNT(*) FROM `axiomatic-jet-425505-j5.modulabs_project.data7`) -
  (SELECT COUNT(*) FROM `axiomatic-jet-425505-j5.modulabs_project.data7_filtered`) AS deleted_count;
```

`Description` 살펴보기

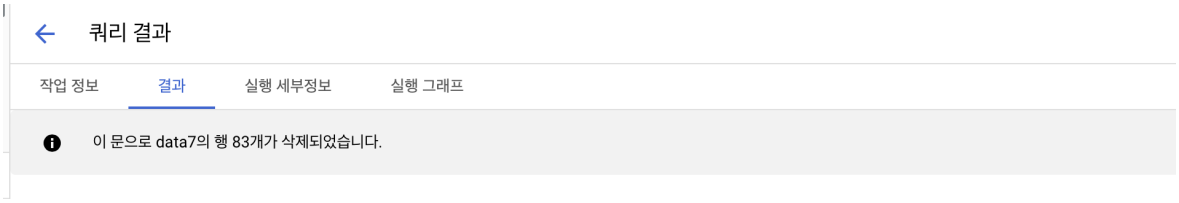
- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS Frequency
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
GROUP BY Description
ORDER BY Frequency DESC
LIMIT 30;
```

행	Description	Frequency
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY ...	1224
8	POSTAGE	1196
9	LUNCH BAG BLACK SKULL.	1099

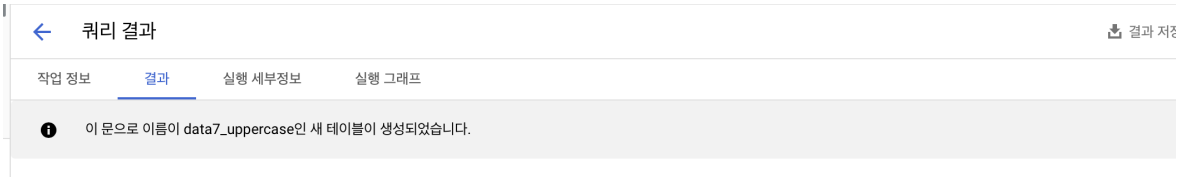
- **서비스 관련 정보를 포함하는 행들을 제거하기**

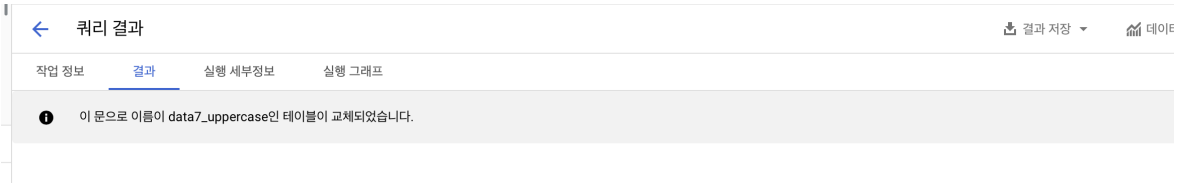
```
DELETE FROM `axiomatic-jet-425505-j5.modulabs_project.data7`
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `axiomatic-jet-425505-j5.modulabs_project.data7_uppercase` AS
SELECT
  InvoiceNo,
  StockCode,
  UPPER(Description) AS Description,
  Quantity,
  InvoiceDate,
  UnitPrice,
  CustomerID,
  Country
FROM `axiomatic-jet-425505-j5.modulabs_project.data7`;

#2
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.data7_uppercase AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM axiomatic-jet-425505-j5.modulabs_project.data7;
```





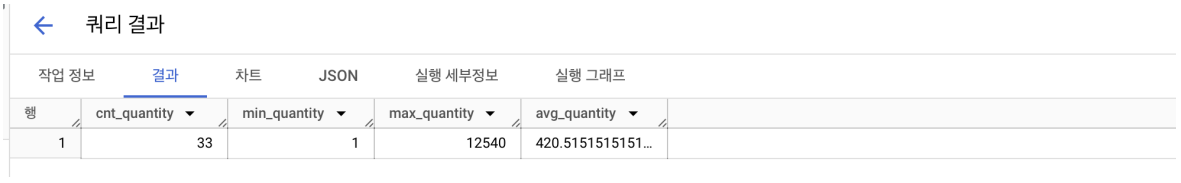
`UnitPrice` 살펴보기

- **`UnitPrice`의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase;
```

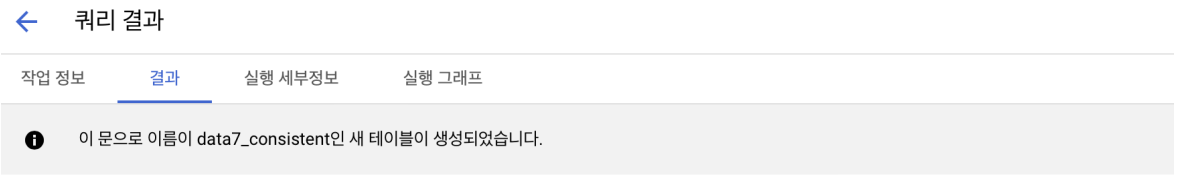
- **단가가 0원인 거래의 개수, 구매 수량(`Quantity`)의 최솟값, 최댓값, 평균 구하기**

```
SELECT
  COUNT(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM `axiomatic-jet-425505-j5.modulabs_project.data`
WHERE UnitPrice = 0;
```



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

```
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.data7_consistent AS
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
WHERE UnitPrice != 0;
```

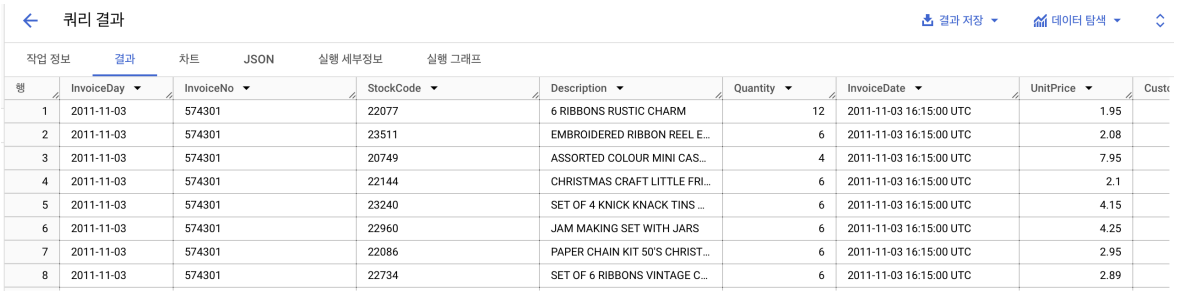


11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
SELECT
  PARSE_DATE('%Y-%m-%d', FORMAT_DATE('%Y-%m-%d', CAST(InvoiceDate AS DATE))) AS InvoiceDay,
  *
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```



쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceDay	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custo
1	2011-11-03	574301	22077	6 RIBBONS RUSTIC CHARM	12	2011-11-03 16:15:00 UTC	1.95	
2	2011-11-03	574301	23511	EMBROIDERED RIBBON REEL E...	6	2011-11-03 16:15:00 UTC	2.08	
3	2011-11-03	574301	20749	ASSORTED COLOUR MINI CAS...	4	2011-11-03 16:15:00 UTC	7.95	
4	2011-11-03	574301	22144	CHRISTMAS CRAFT LITTLE FRI...	6	2011-11-03 16:15:00 UTC	2.1	
5	2011-11-03	574301	23240	SET OF 4 KNICK KNACK TINS ...	6	2011-11-03 16:15:00 UTC	4.15	
6	2011-11-03	574301	22960	JAM MAKING SET WITH JARS	6	2011-11-03 16:15:00 UTC	4.25	
7	2011-11-03	574301	22086	PAPER CHAIN KIT 50'S CHRIST...	6	2011-11-03 16:15:00 UTC	2.95	
8	2011-11-03	574301	22734	SET OF 6 RIBBONS VINTAGE C...	6	2011-11-03 16:15:00 UTC	2.89	

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  MAX(PARSE_DATE('%Y-%m-%d', FORMAT_DATE('%Y-%m-%d', CAST(InvoiceDate AS DATE)))) AS most_recent_pur
FROM `axiomatic-jet-425505-j5.modulabs_project.data`;
```

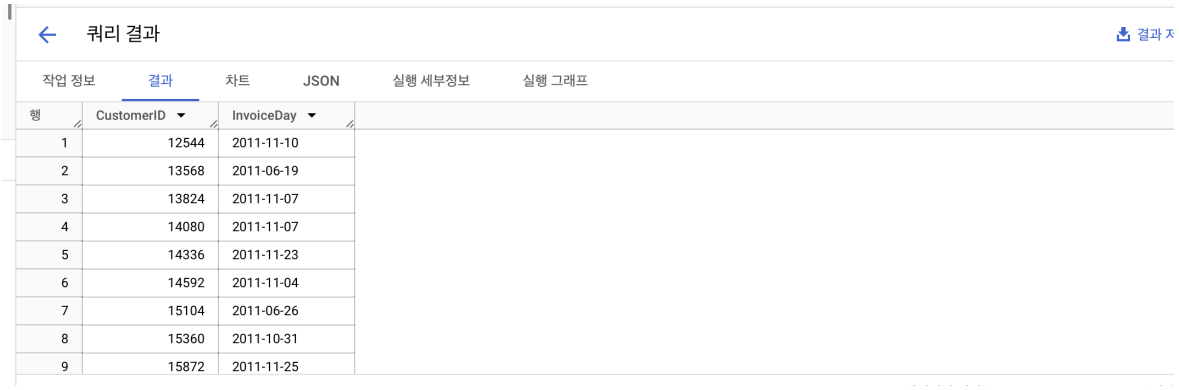
쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	most_recent_purcha
1	2011-12-09

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

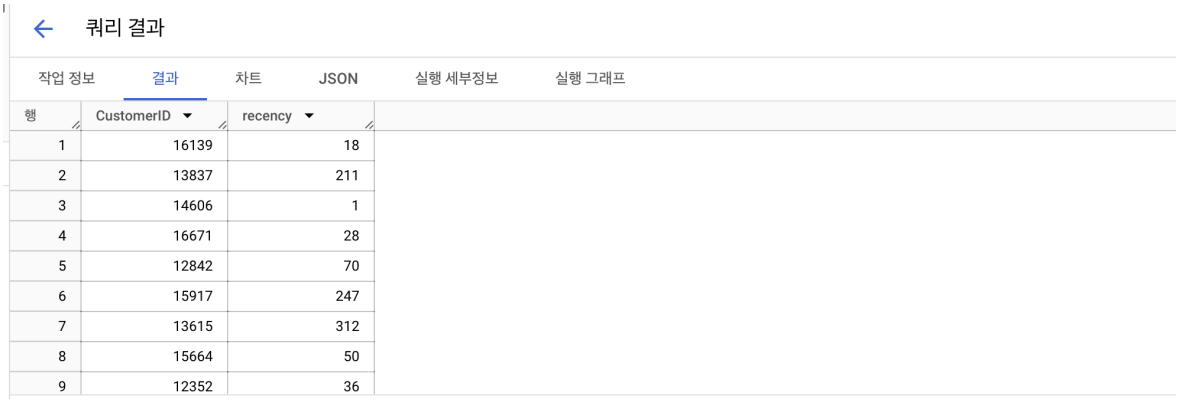
```
SELECT
    CustomerID,
    MAX(PARSE_DATE('%Y-%m-%d', FORMAT_DATE('%Y-%m-%d', CAST(InvoiceDate AS DATE)))) AS InvoiceDay
FROM `axiomatic-jet-425505-j5.modulabs_project.data5`
GROUP BY CustomerID;
```



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
9	15872	2011-11-25

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

```
SELECT
  CustomerID,
  DATE_DIFF(MAX(InvoiceDay) OVER (), InvoiceDay, DAY) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  GROUP BY CustomerID
);
```

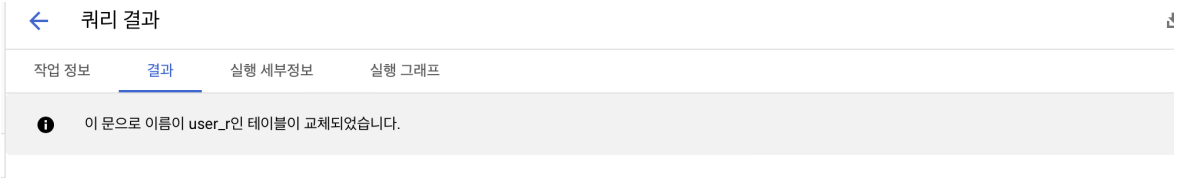


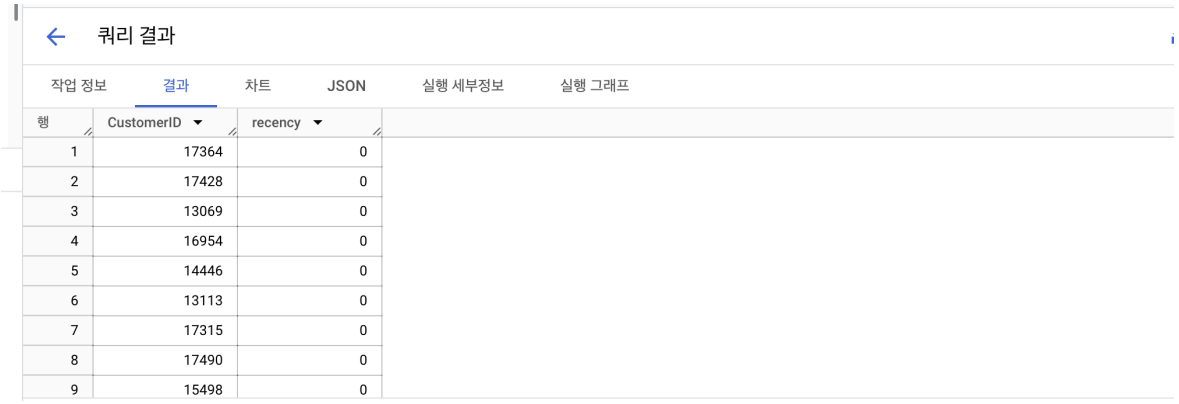
행	CustomerID	recency
1	16139	18
2	13837	211
3	14606	1
4	16671	28
5	12842	70
6	15917	247
7	13615	312
8	15664	50
9	12352	36

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

```
# InvoiceDate 컬럼을 연월일 자료형으로 변경하여 새로운 테이블 생성
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.data7_cleaned AS
SELECT
  PARSE_DATE('%Y-%m-%d', FORMAT_DATE('%Y-%m-%d', CAST(InvoiceDate AS DATE))) AS InvoiceDay,
  InvoiceNo,
  StockCode,
  UPPER(Description) AS Description,
  Quantity,
  UnitPrice,
  CustomerID,
  Country
FROM
  axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
WHERE
```

```
  UnitPrice != 0;

# 가장 최근 구매 일자 찾기
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.most_recent_purchase_date AS
SELECT
  MAX(InvoiceDay) AS most_recent_purchase_date
FROM
  axiomatic-jet-425505-j5.modulabs_project.data7_cleaned;

# 유저별 가장 큰 InvoiceDay 찾기
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.customer_last_purchase AS
SELECT
  CustomerID,
  MAX(InvoiceDay) AS InvoiceDay
FROM
  axiomatic-jet-425505-j5.modulabs_project.data7_cleaned
GROUP BY
  CustomerID;

# 가장 최근 일자와 유저별 마지막 구매일 간 차이 계산하여 최종 데이터셋 생성
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.user_r AS
SELECT
  clp.CustomerID,
  DATE_DIFF(mrd.most_recent_purchase_date, clp.InvoiceDay, DAY) AS recency
FROM
  axiomatic-jet-425505-j5.modulabs_project.customer_last_purchase clp,
  axiomatic-jet-425505-j5.modulabs_project.most_recent_purchase_date mrd;


# 완성 테이블 조회
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_r;
```





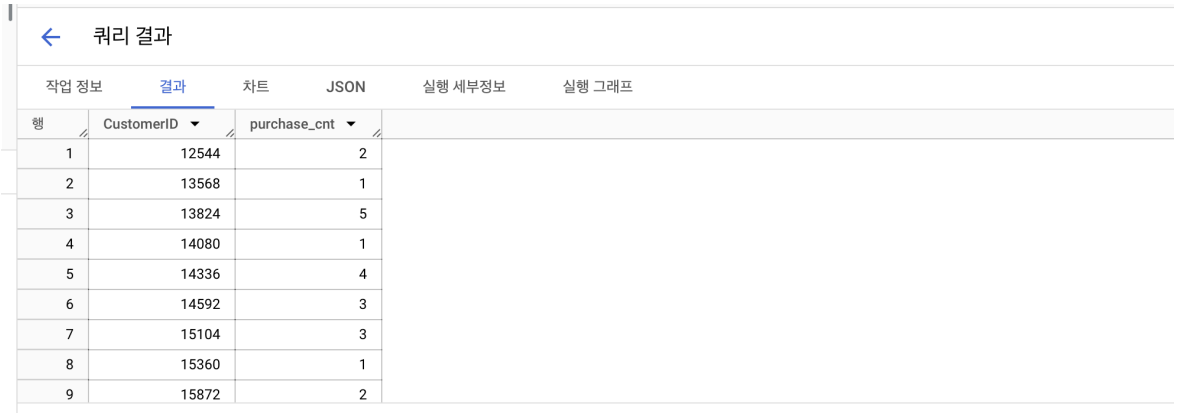
행	CustomerID	recency
1	17364	0
2	17428	0
3	13069	0
4	16954	0
5	14446	0
6	13113	0
7	17315	0
8	17490	0
9	15498	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
```

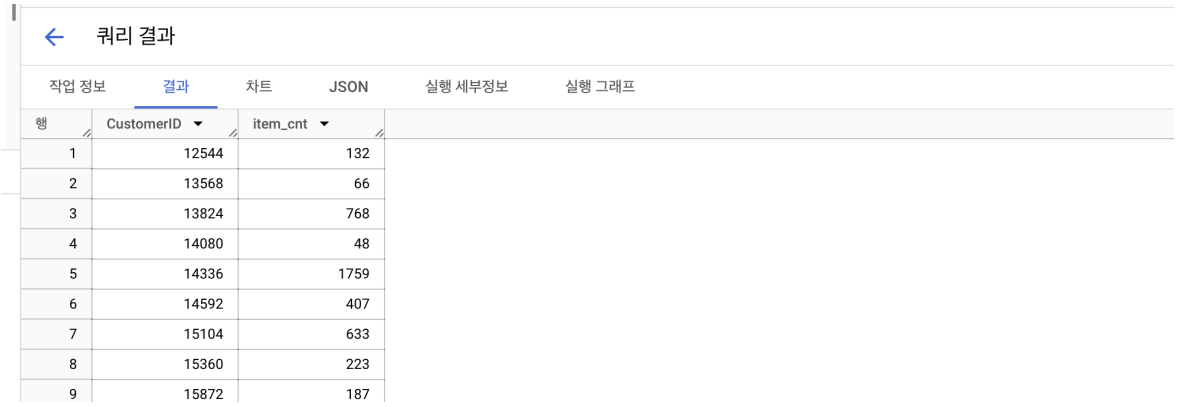
```
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
GROUP BY CustomerID;
```



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
9	15872	2

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
GROUP BY CustomerID;
```

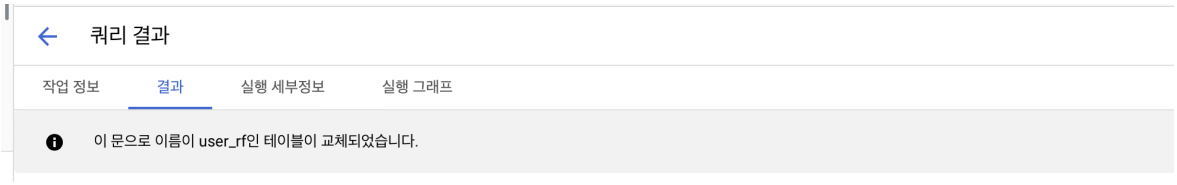


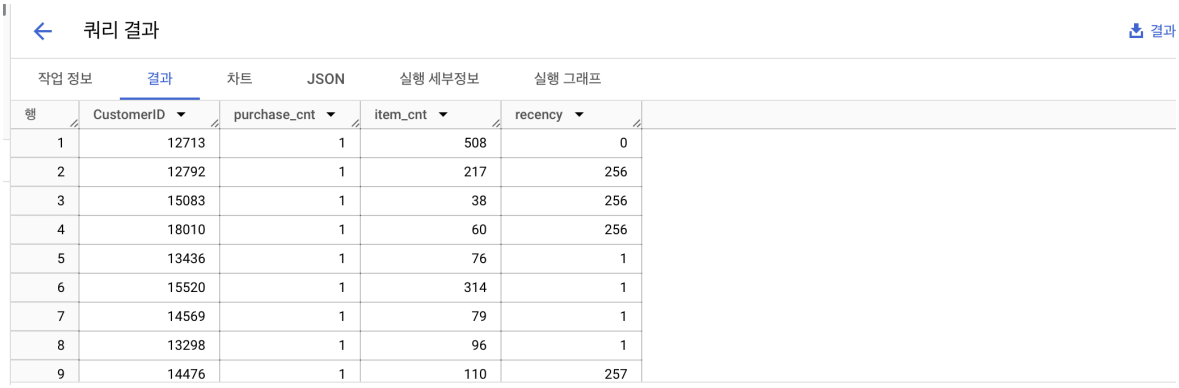
행	CustomerID	item_cnt
1	12544	132
2	13568	66
3	13824	768
4	14080	48
5	14336	1759
6	14592	407
7	15104	633
8	15360	223
9	15872	187

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

```
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
```

```
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  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 axiomatic-jet-425505-j5.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;

#테이블 조회
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_rf;
```



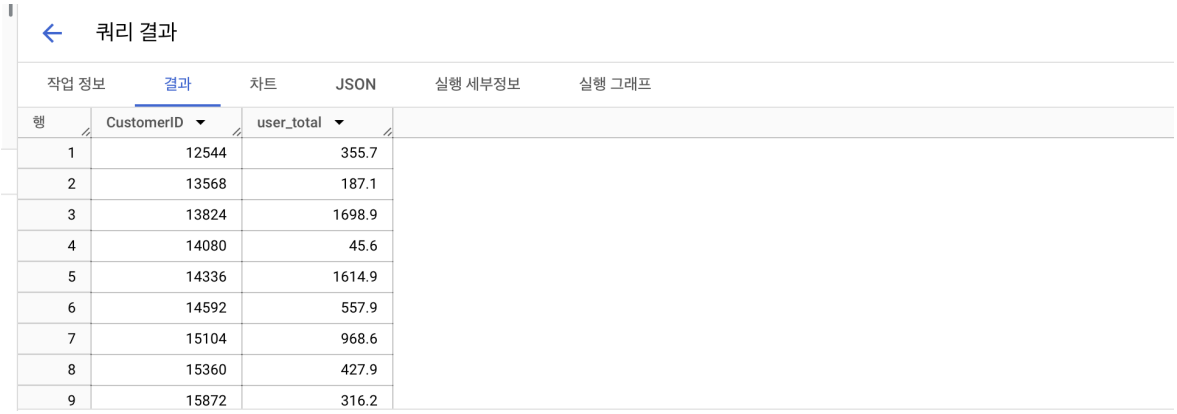


행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	508	0
2	12792	1	217	256
3	15083	1	38	256
4	18010	1	60	256
5	13436	1	76	1
6	15520	1	314	1
7	14569	1	79	1
8	13298	1	96	1
9	14476	1	110	257

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

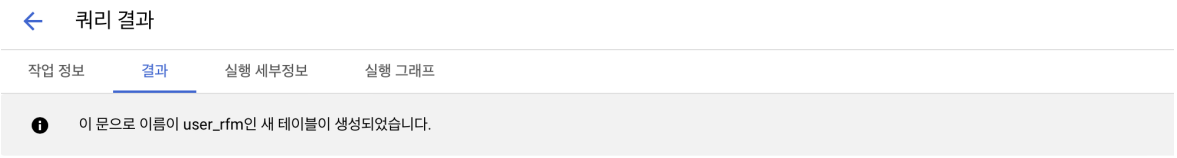
```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
GROUP BY CustomerID;
```

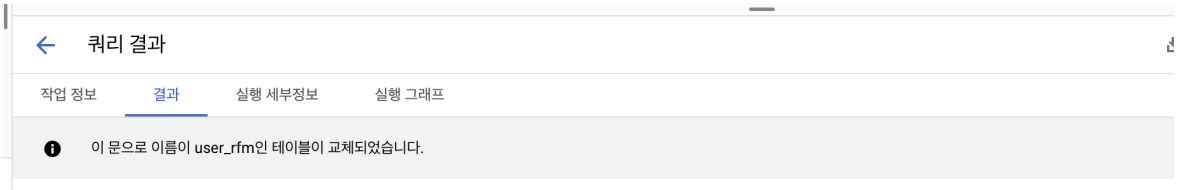


행	CustomerID	user_total
1	12544	355.7
2	13568	187.1
3	13824	1698.9
4	14080	45.6
5	14336	1614.9
6	14592	557.9
7	15104	968.6
8	15360	427.9
9	15872	316.2

- **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 axiomatic-jet-425505-j5.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 axiomatic-jet-425505-j5.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

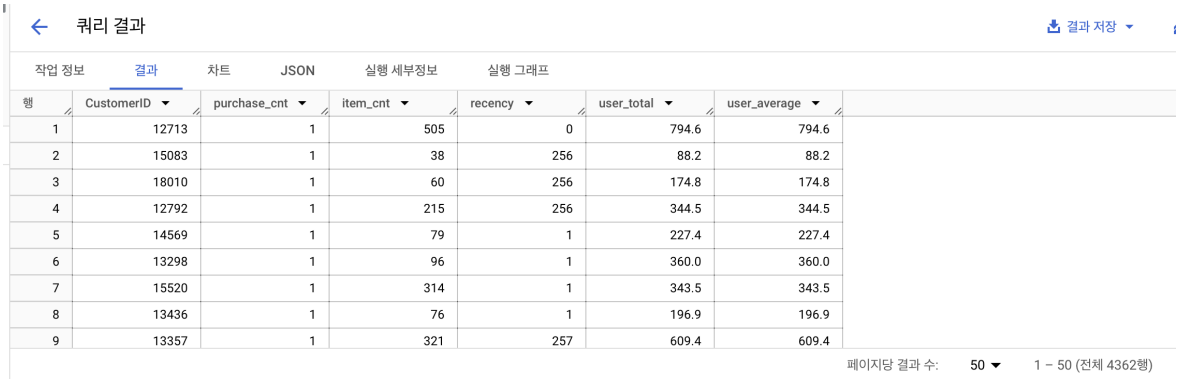




RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM axiomatic-jet-425505-j5.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	12713	1	505	0	794.6	794.6
2	15083	1	38	256	88.2	88.2
3	18010	1	60	256	174.8	174.8
4	12792	1	215	256	344.5	344.5
5	14569	1	79	1	227.4	227.4
6	13298	1	96	1	360.0	360.0
7	15520	1	314	1	343.5	343.5
8	13436	1	76	1	196.9	196.9
9	13357	1	321	257	609.4	609.4

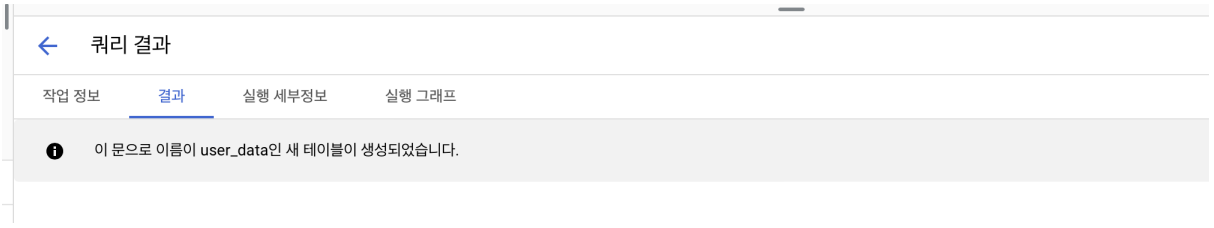
페이지당 결과 수: 50 1 – 50 (전체 4362행)

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**

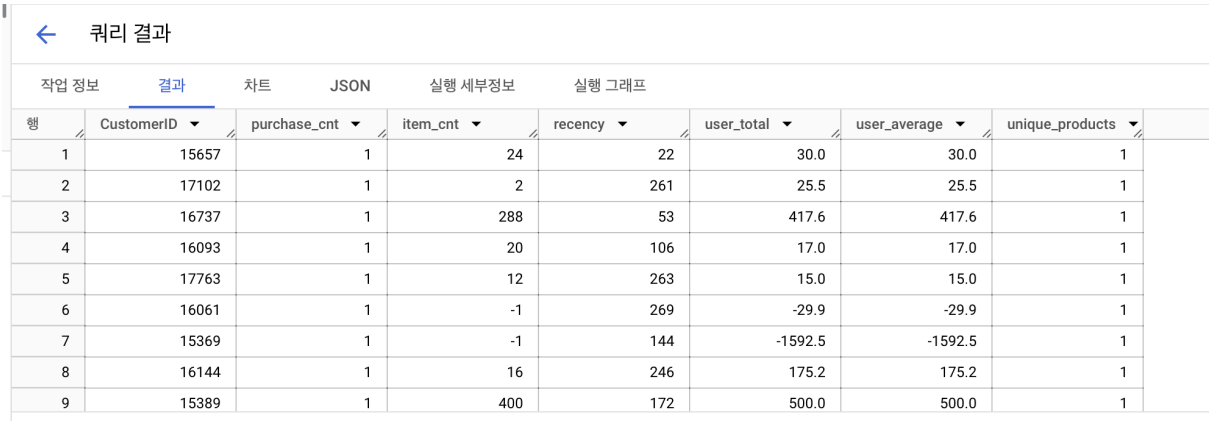
```
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM axiomatic-jet-425505-j5.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;

# 테이블 조회
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_data;
```



쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.



쿼리 결과

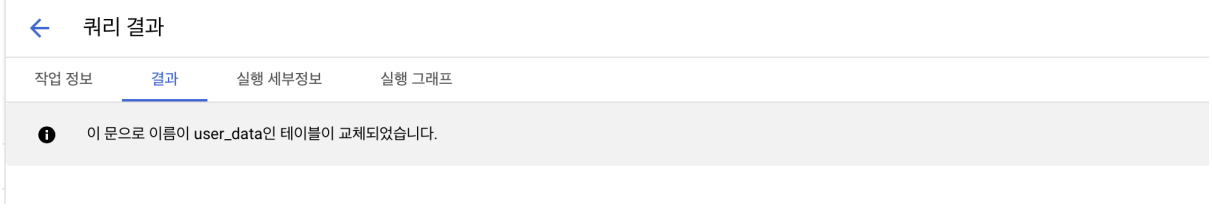
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보 | 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15657	1	24	22	30.0	30.0	1
2	17102	1	2	261	25.5	25.5	1
3	16737	1	288	53	417.6	417.6	1
4	16093	1	20	106	17.0	17.0	1
5	17763	1	12	263	15.0	15.0	1
6	16061	1	-1	269	-29.9	-29.9	1
7	15369	1	-1	144	-1592.5	-1592.5	1
8	16144	1	16	246	175.2	175.2	1
9	15389	1	400	172	500.0	500.0	1

2. 평균 구매 주기

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

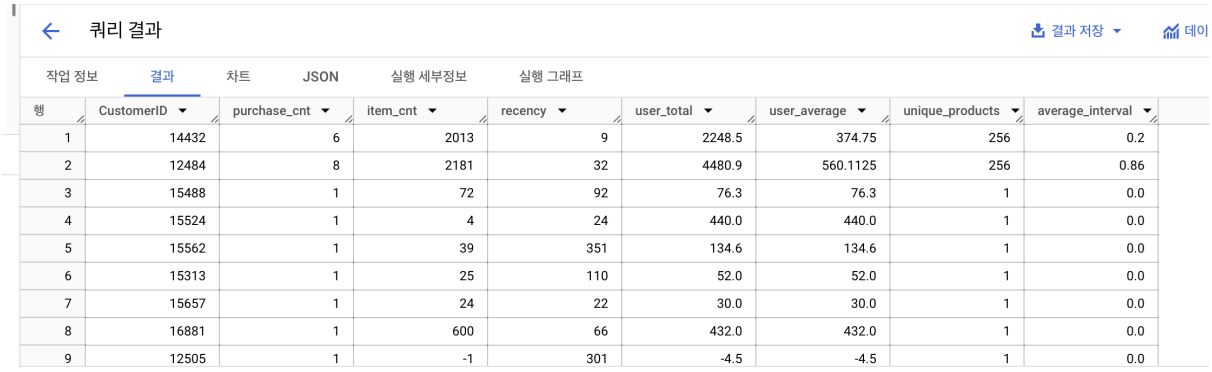
```
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.user_data AS
WITH purchase_intervals AS (
  -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    FROM
      axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM axiomatic-jet-425505-j5.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

# 결과 테이블 조회
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_data;
```



쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 user_data인 테이블이 교체되었습니다.

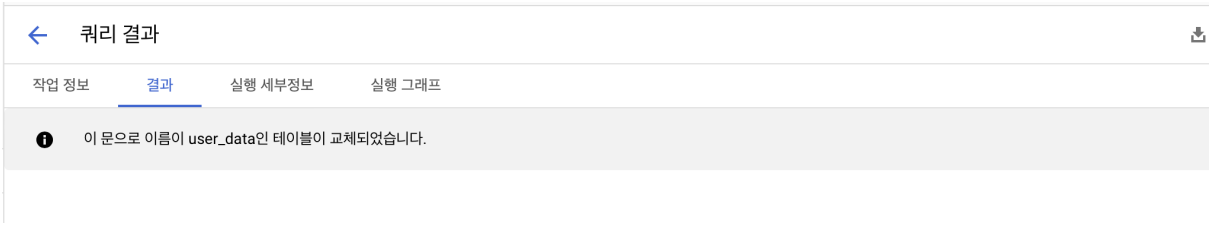


행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	14432	6	2013	9	2248.5	374.75	256	0.2
2	12484	8	2181	32	4480.9	560.1125	256	0.86
3	15488	1	72	92	76.3	76.3	1	0.0
4	15524	1	4	24	440.0	440.0	1	0.0
5	15562	1	39	351	134.6	134.6	1	0.0
6	15313	1	25	110	52.0	52.0	1	0.0
7	15657	1	24	22	30.0	30.0	1	0.0
8	16881	1	600	66	432.0	432.0	1	0.0
9	12505	1	-1	301	-4.5	-4.5	1	0.0

3. 구매 취소 경향성

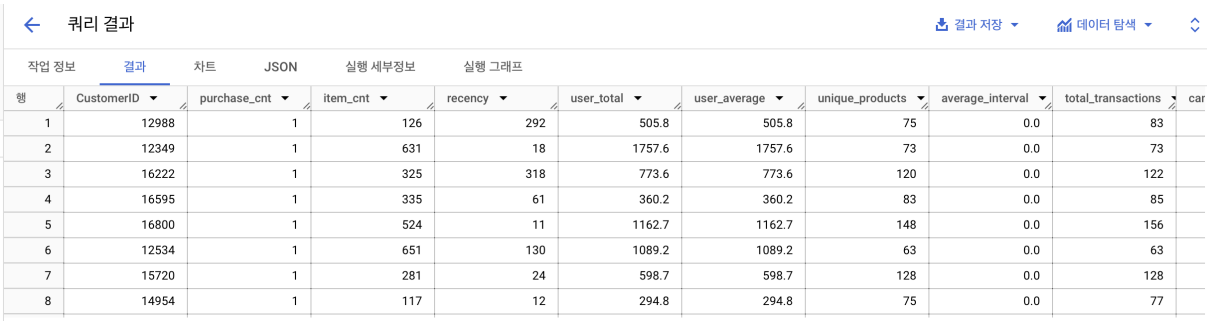
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
    - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
      (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE axiomatic-jet-425505-j5.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 END) AS cancel_frequency
  FROM axiomatic-jet-425505-j5.modulabs_project.data7_uppercase
  GROUP BY CustomerID
)

SELECT u.*,
       t.total_transactions,
       t.cancel_frequency,
       ROUND(CAST(t.cancel_frequency AS FLOAT64) / t.total_transactions, 2) AS cancel_rate
FROM axiomatic-jet-425505-j5.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```



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

```
SELECT *
FROM axiomatic-jet-425505-j5.modulabs_project.user_data;
```



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	average_interval	total_transactions	car
1	12988	1	126	292	505.8	505.8	75	0.0	83	
2	12349	1	631	18	1757.6	1757.6	73	0.0	73	
3	16222	1	325	318	773.6	773.6	120	0.0	122	
4	16595	1	335	61	360.2	360.2	83	0.0	85	
5	16800	1	524	11	1162.7	1162.7	148	0.0	156	
6	12534	1	651	130	1089.2	1089.2	63	0.0	63	
7	15720	1	281	24	598.7	598.7	128	0.0	128	
8	14954	1	117	12	294.8	294.8	75	0.0	77	

👉 SQL 쿼리문을 통해 얻은 결과 해석/인사이트

- 활발한 고객 식별
  - purchase_cnt와 item_cnt가 높고, user_total이 높은 고객은 회사의 주요 고객 가능성 유망
  - CustomerID 12688은 3,028개 아이템 구매, 총 지출액이 4,873.8로 매우 활발하게 구매를 진행
- 휴면 고객 식별
  - recency 값이 높은 고객은 유도 캠페인 대상으로 적절
  - CustomerID 12988은 292전이 마지막 구매일로 오랫동안 휴면상태, 캠페인 대상 범위에 적절

  → 특별 할인, 프로모션 제공
- 다양한 상품 구매
  - unique_products가 높은 고객은 다양한 제품 추천에 긍정적 반응 가능성이 높음
  - CustomerID 16900dms 148개의 고유한 상품을 구매 → 고객이 비교적 다양한 제품에 관심있음
- 취소 빈도와 비율
  - 실제 데이터에서 cancel_rate가 높은 고객들은 거래 신뢰성이 낮음
  - cancel_rate가 높은 고객은 cancel_frequency와 cancel_rate 값이 0

  → cancel 이슈 파악을 위한 지원을 제공해 고객 만족도 상승 필요
- 평균 거래 금액
  - user_average가 높은 고객들은 프리미엄 고객으로 분류
  - CustomerID 12688의 user_average는 4,873.8로 타 고객에 비해 매우 높은 평균 거래 금액

  → CustomerID 12688의 경우 user_average(purchase_cnt, item_cnt, user_total)이 높아 프리미엄 고객으로 분류하여 추가 혜택 제공이 필요