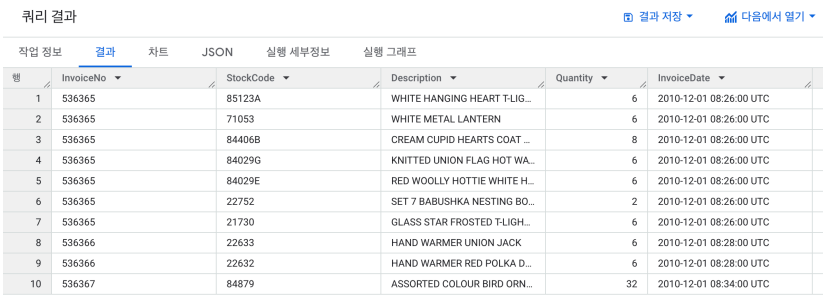
04.09

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `modulabs_project.data`
LIMIT 10
```



쿼리 결과

결과 저장 ▾ 다음에서 열기 ▾

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

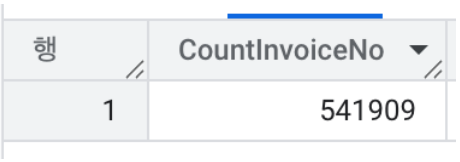
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC
4	536365	84029G	KNITTED UNION FLAG HOT WA...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

[결과 이미지를 넣어주세요]

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(InvoiceNo) AS CountInvoiceNo
FROM `modulabs_project.data`
```

[결과 이미지를 넣어주세요]

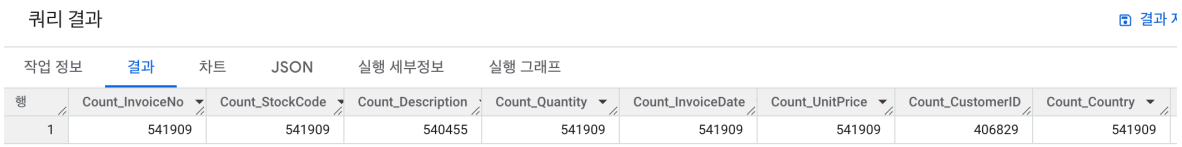


행	CountInvoiceNo
1	541909

데이터 수 세기

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

```
SELECT
 COUNT(InvoiceNo) AS Count_InvoiceNo,
 COUNT(StockCode) AS Count_StockCode,
 COUNT(Description) AS Count_Description,
 COUNT(Quantity) AS Count_Quantity,
 COUNT(InvoiceDate) AS Count_InvoiceDate,
 COUNT(UnitPrice) AS Count_UnitPrice,
 COUNT(CustomerID) AS Count_CustomerID,
 COUNT(Country) AS Count_Country
FROM `modulabs_project.data`
```



[결과 이미지를 넣어주세요]

11-4. 데이터 전처리 방법(1): 결측치 제거

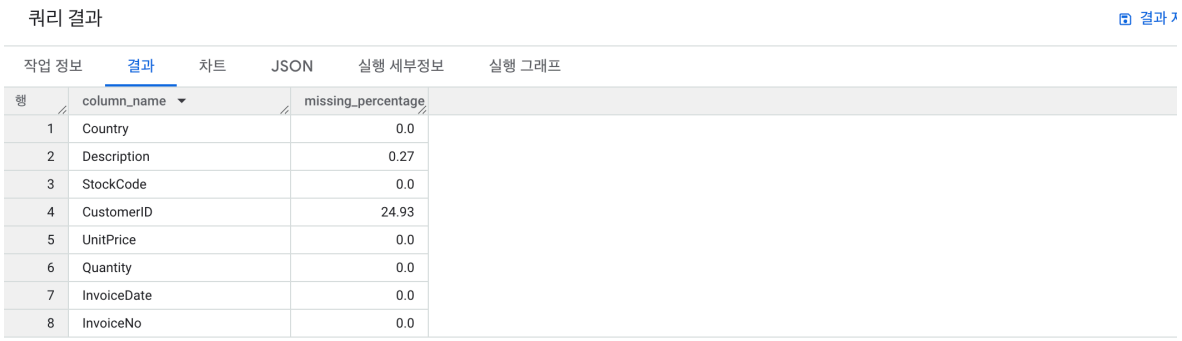
컬럼 별 누락된 값의 비율 계산

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

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

[결과 이미지를 넣어주세요]



쿼리 결과

결과 저

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

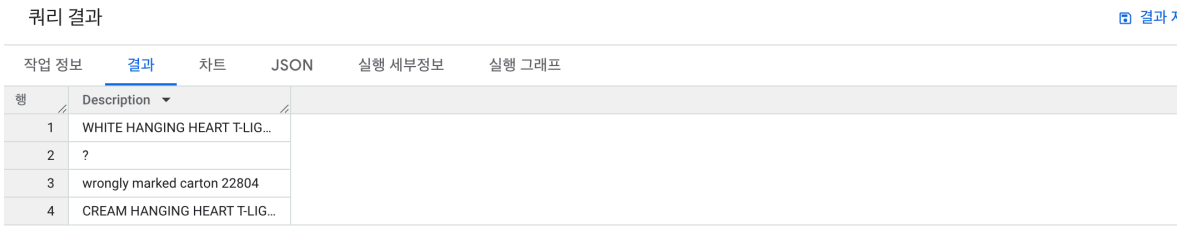
행	column_name	missing_percentage
1	Country	0.0
2	Description	0.27
3	StockCode	0.0
4	CustomerID	24.93
5	UnitPrice	0.0
6	Quantity	0.0
7	InvoiceDate	0.0
8	InvoiceNo	0.0

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 저

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

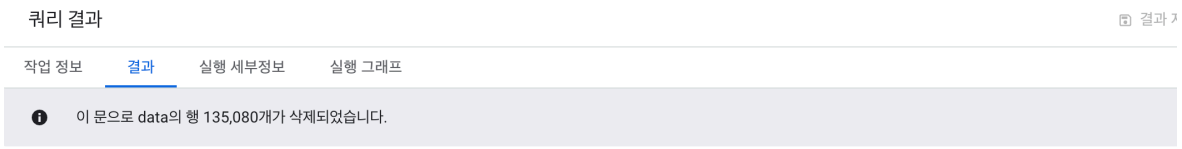
행	Description
1	WHITE HANGING HEART T-LIG...
2	?
3	wrongly marked carton 22804
4	CREAM HANGING HEART T-LIG...

결측치 처리

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

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

[결과 이미지를 넣어주세요]



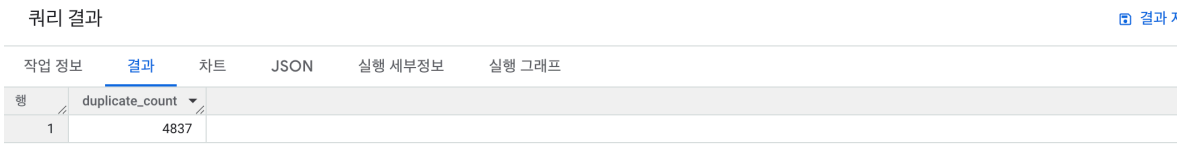
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

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

[결과 이미지를 넣어주세요]

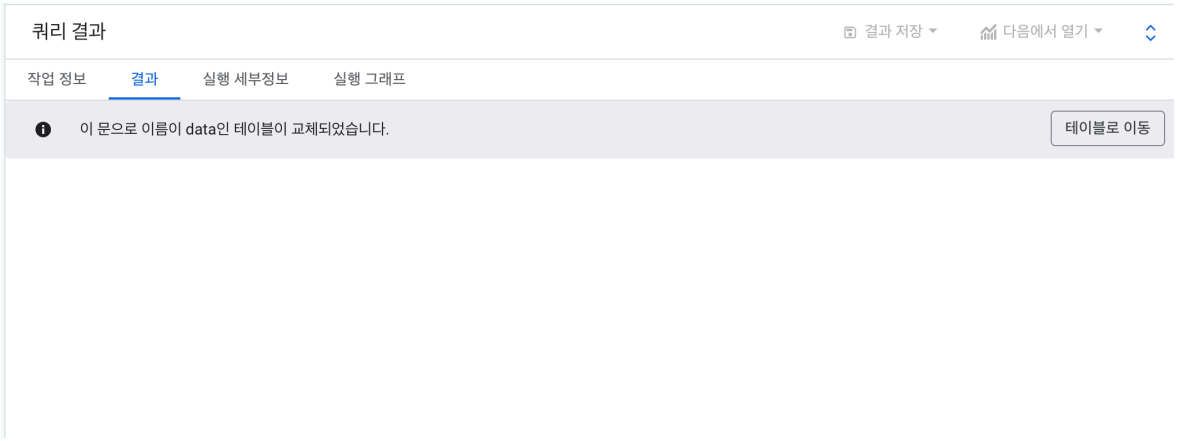


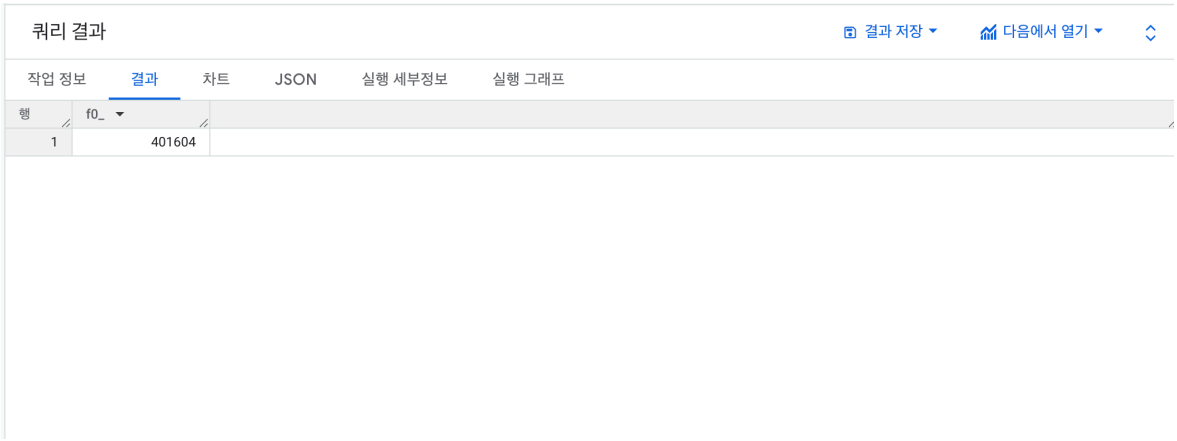
중복값 처리

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

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT DISTINCT *
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]





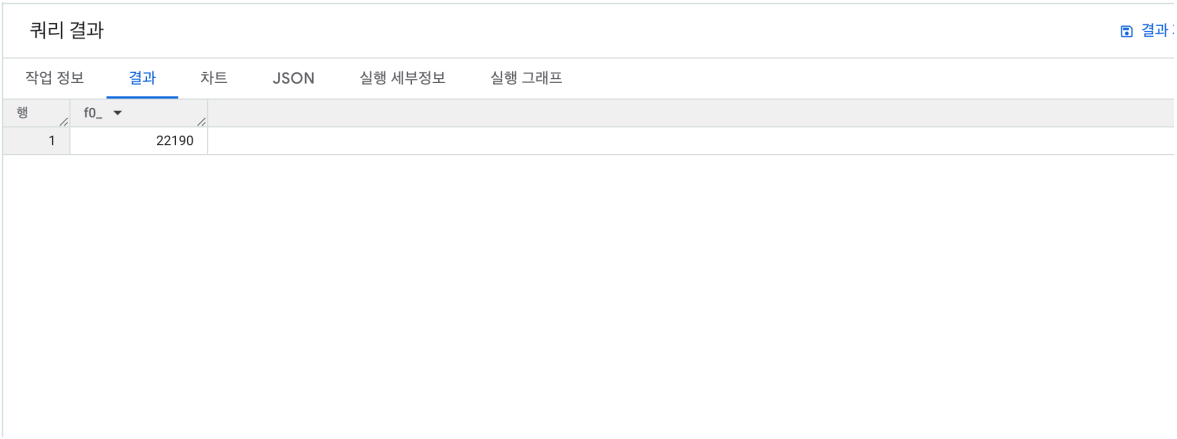
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM modulabs_project.data
```

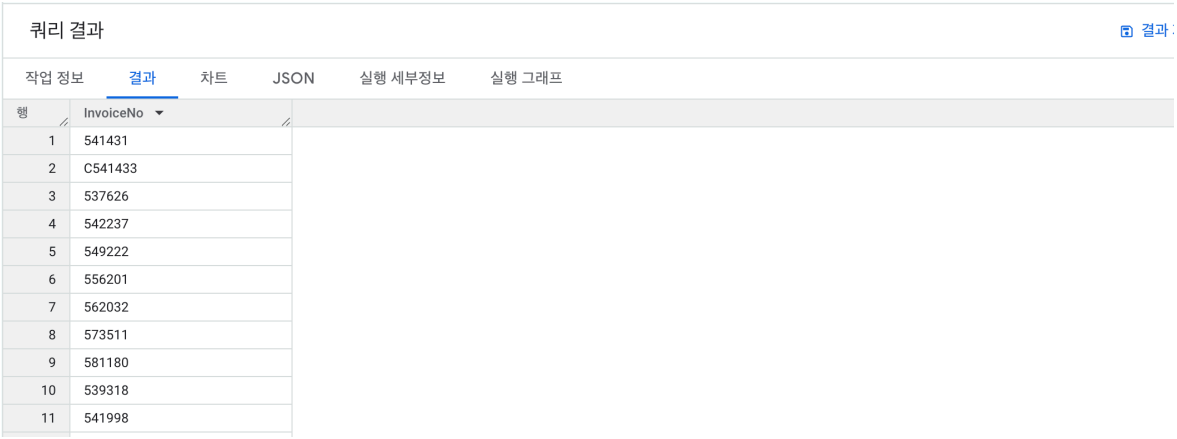
[결과 이미지를 넣어주세요]



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

```
SELECT DISTINCT InvoiceNo
FROM modulabs_project.data
LIMIT 100
```

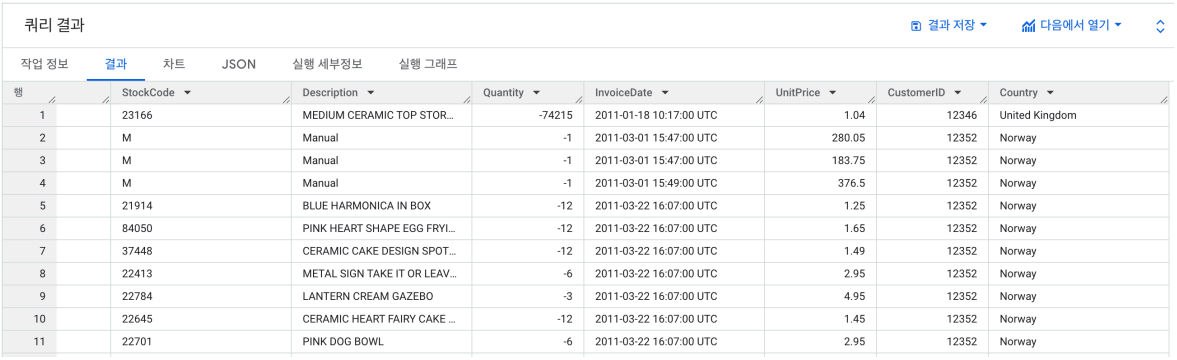
[결과 이미지를 넣어주세요]



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

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

[결과 이미지를 넣어주세요]



쿼리 결과 — 결과 저장 — 다음에서 열기

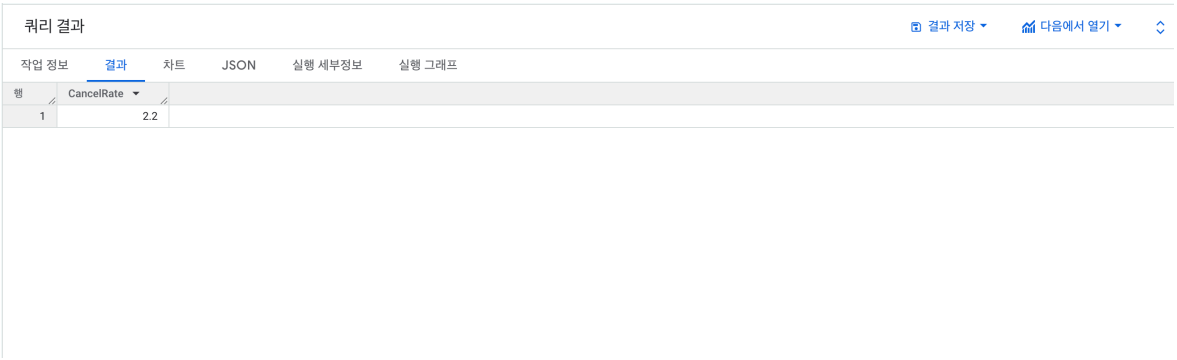
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보 | 실행 그래프

행		StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1		23166	MEDIUM CERAMIC TOP STOR…	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2		M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3		M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4		M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5		21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
6		84050	PINK HEART SHAPE EGG FRYI…	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
7		37448	CERAMIC CAKE DESIGN SPOT…	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
8		22413	METAL SIGN TAKE IT OR LEAV…	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
9		22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
10		22645	CERAMIC HEART FAIRY CAKE …	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
11		22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway

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

```
SELECT
    ROUND((SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 END)*100) / COUNT(*), 1) AS CancelRate
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]



쿼리 결과 — 결과 저장 — 다음에서 열기

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

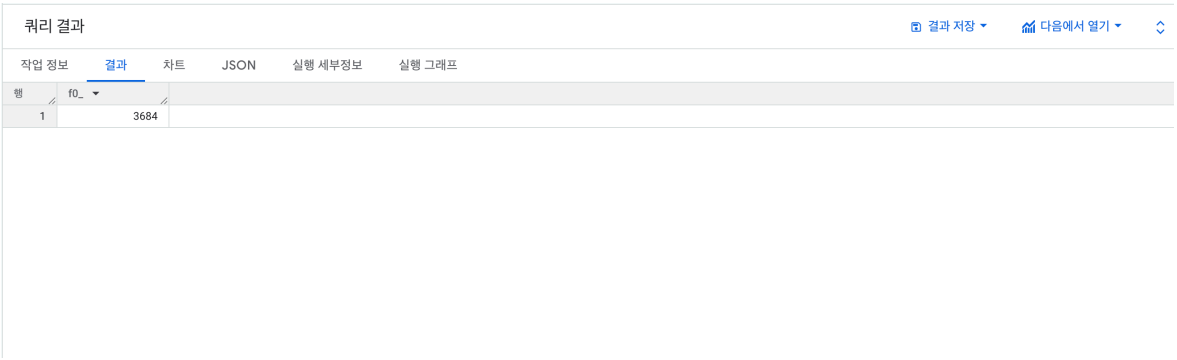
행	CancelRate
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode)
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]



쿼리 결과 — 결과 저장 — 다음에서 열기

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

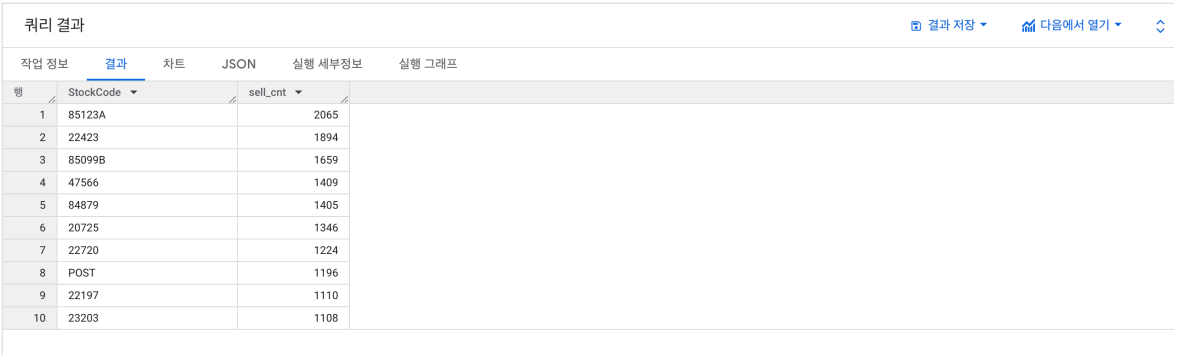
행	f0_
1	3684

- **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.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

[결과 이미지를 넣어주세요]



쿼리 결과 — 결과 저장 — 다음에서 열기

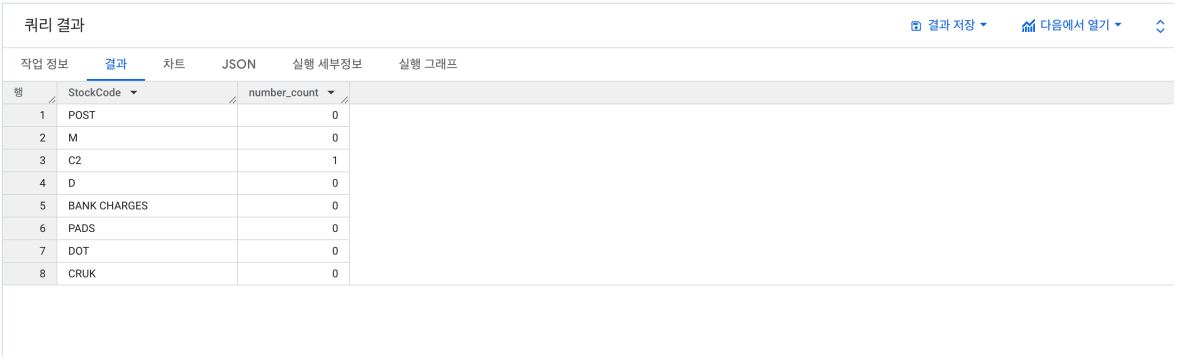
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보 | 실행 그래프

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
9	22197	1110
10	23203	1108

- `StockCode`**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값**들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM modulabs_project.data
)
WHERE number_count = 0 or number_count = 1
```

[결과 이미지를 넣어주세요]



쿼리 결과 — 결과 저장 — 다음에서 열기

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

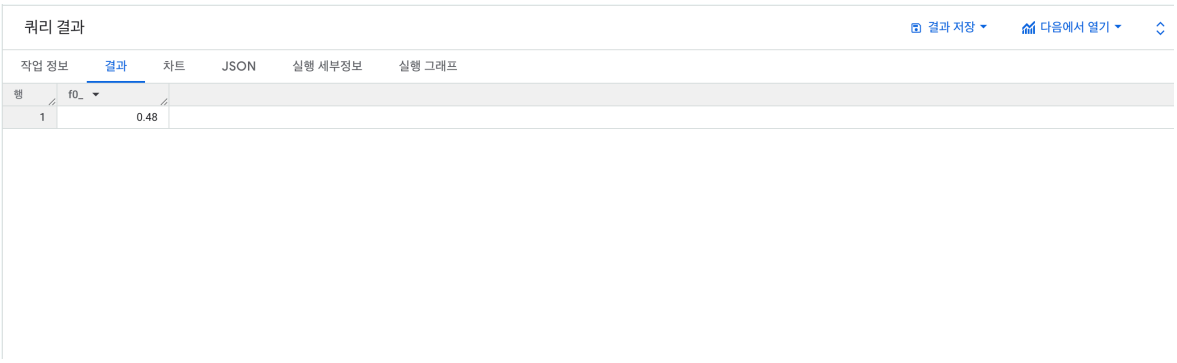
행	StockCode	number_count
1	POST	0
2	M	0
3	C2	1
4	D	0
5	BANK CHARGES	0
6	PADS	0
7	DOT	0
8	CRUK	0

- `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 = 0 or number_count = 1 THEN 1 END)*100) / COUNT(*), 2)
FROM (
```

```
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM modulabs_project.data
)
```

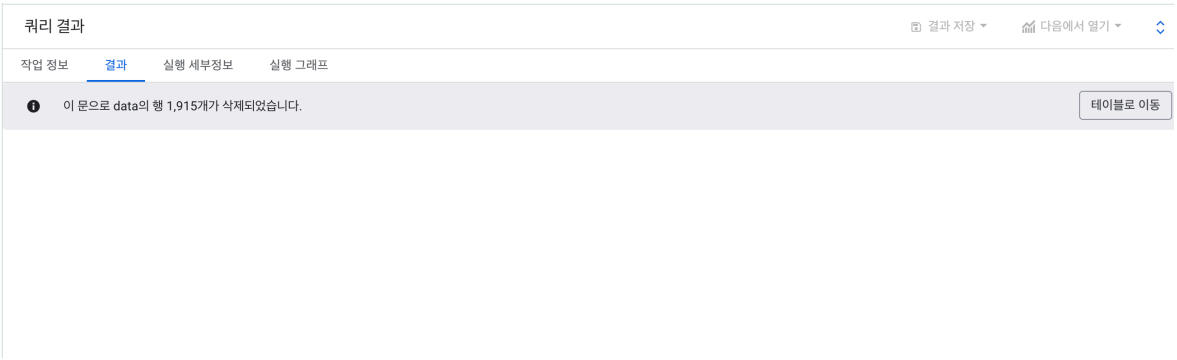
[결과 이미지를 넣어주세요]



- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM 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 modulabs_project.data)
   WHERE number_count = 0 or number_count = 1
)
```

[결과 이미지를 넣어주세요]

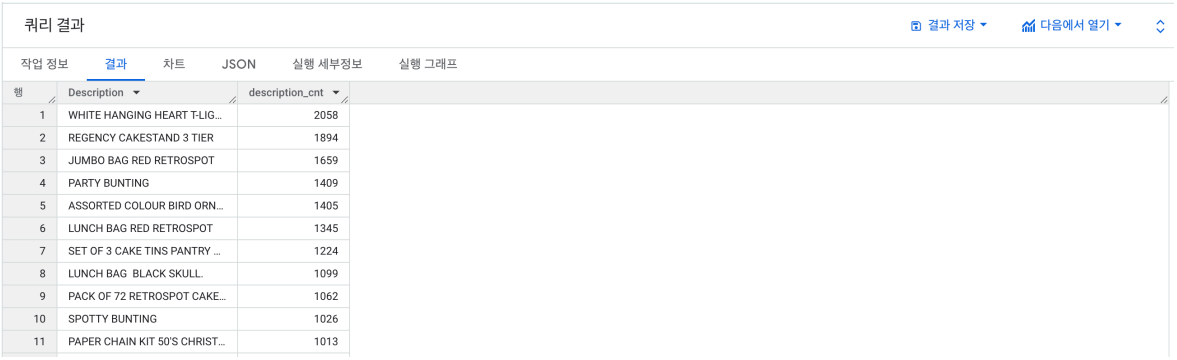


Description 살펴보기

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

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

[결과 이미지를 넣어주세요]

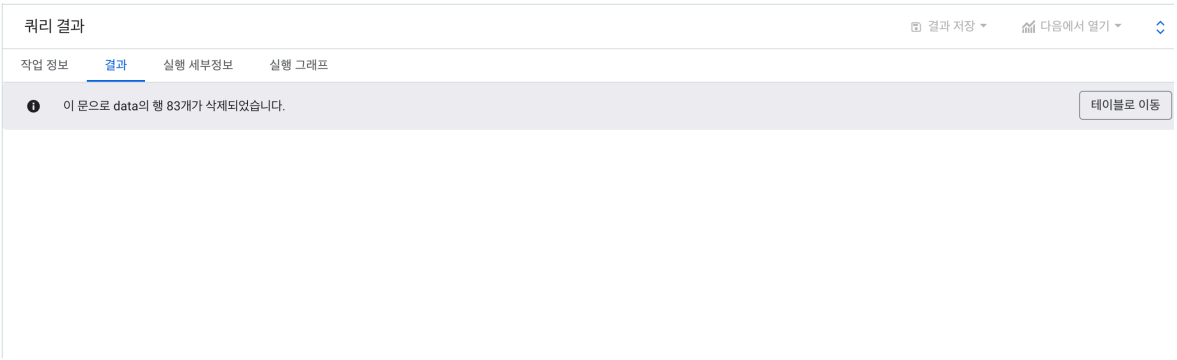


행	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY ...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE...	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013

- **서비스 관련 정보를 포함하는 행들을 제거하기**

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

[결과 이미지를 넣어주세요]



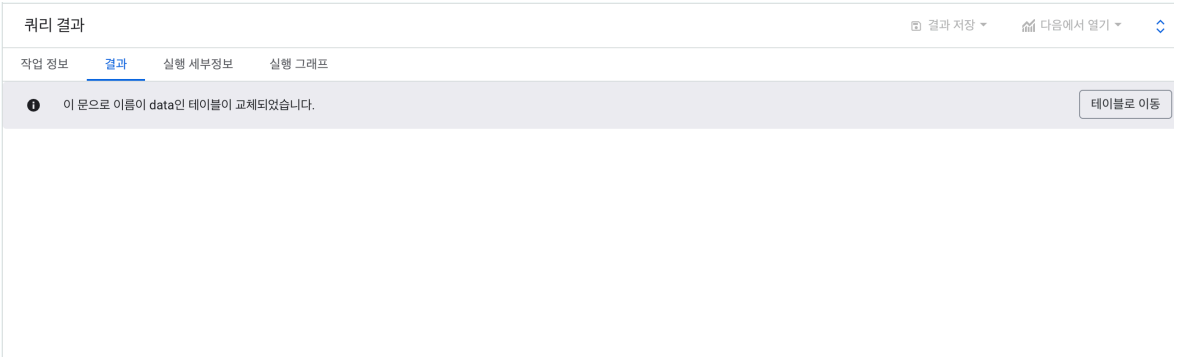

```
-- cf)삭제 행 카운트 방법
DELETE FROM `modulabs_project.data`
WHERE ....

select count(*)
from `modulabs_project.data`
WHERE ,,,,
```

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

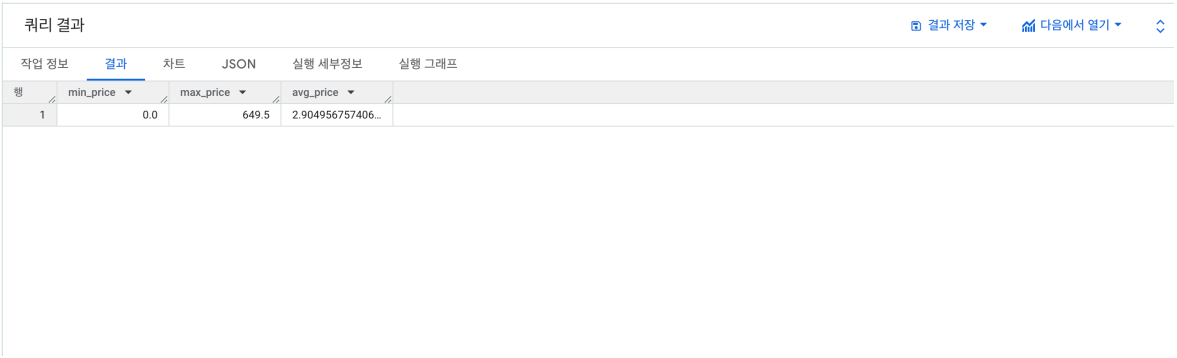


UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

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

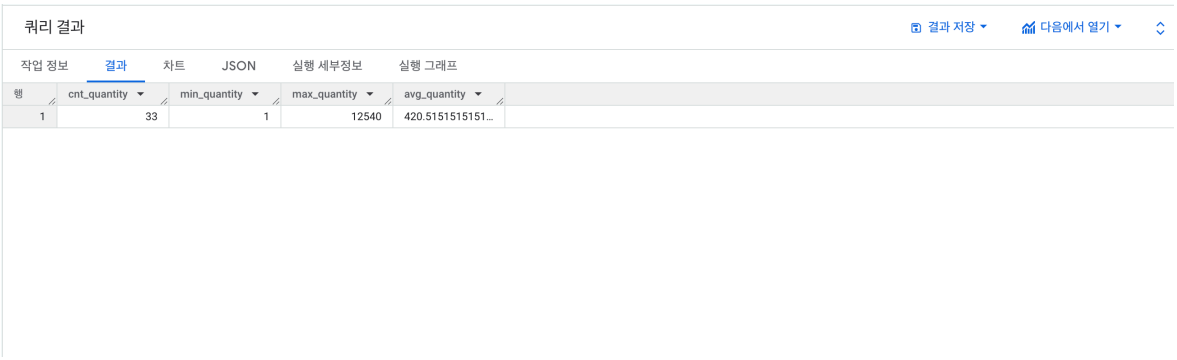
[결과 이미지를 넣어주세요]



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

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

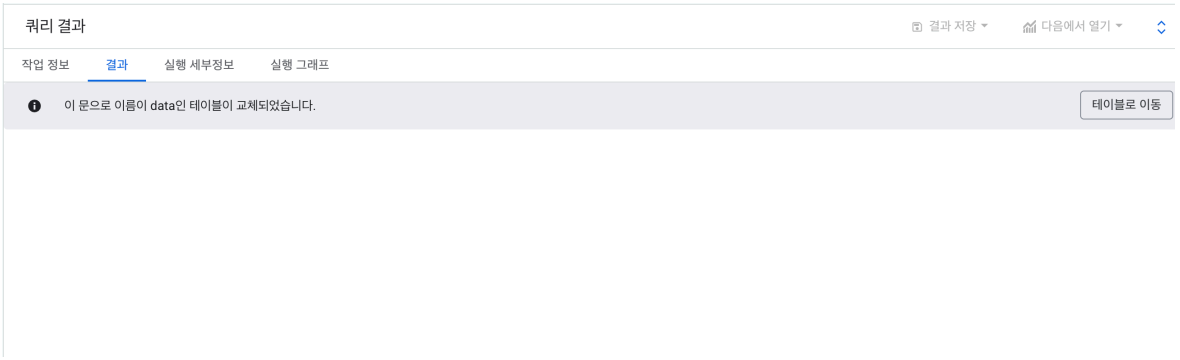
[결과 이미지를 넣어주세요]



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

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

[결과 이미지를 넣어주세요]



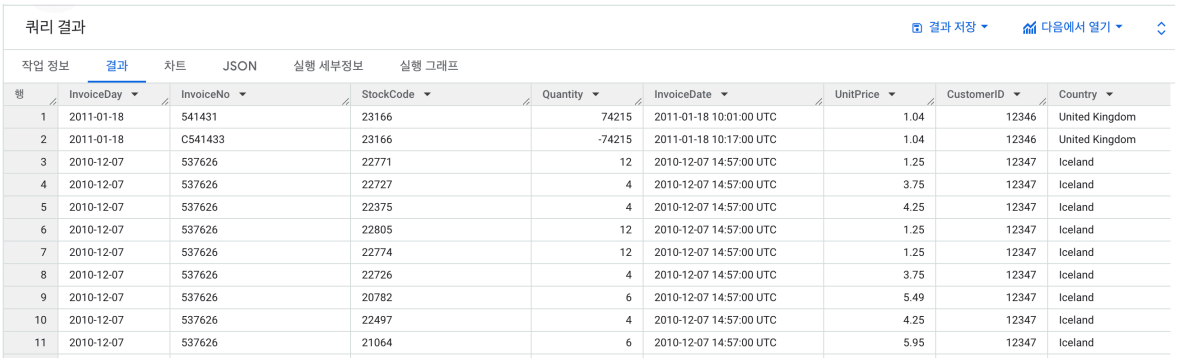
11-7. RFM 스코어

Recency

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

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

[결과 이미지를 넣어주세요]

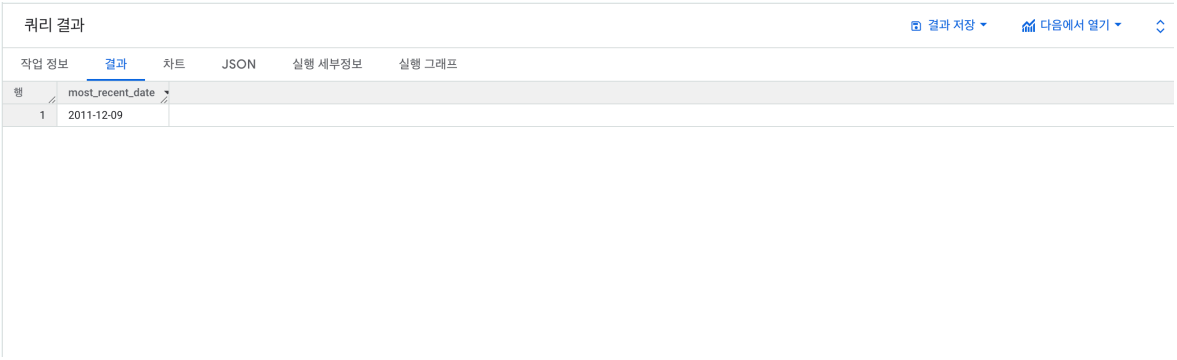


행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
3	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
4	2010-12-07	537626	22727	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
5	2010-12-07	537626	22375	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland
6	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
7	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
8	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
9	2010-12-07	537626	20782	6	2010-12-07 14:57:00 UTC	5.49	12347	Iceland
10	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland
11	2010-12-07	537626	21064	6	2010-12-07 14:57:00 UTC	5.95	12347	Iceland

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(DATE(InvoiceDate)) AS most_recent_date
FROM modulabs_project.data
```

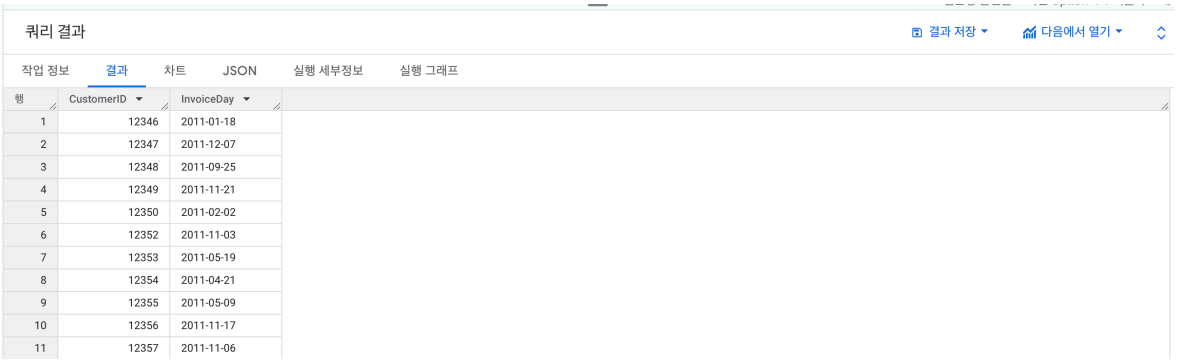
[결과 이미지를 넣어주세요]



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

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

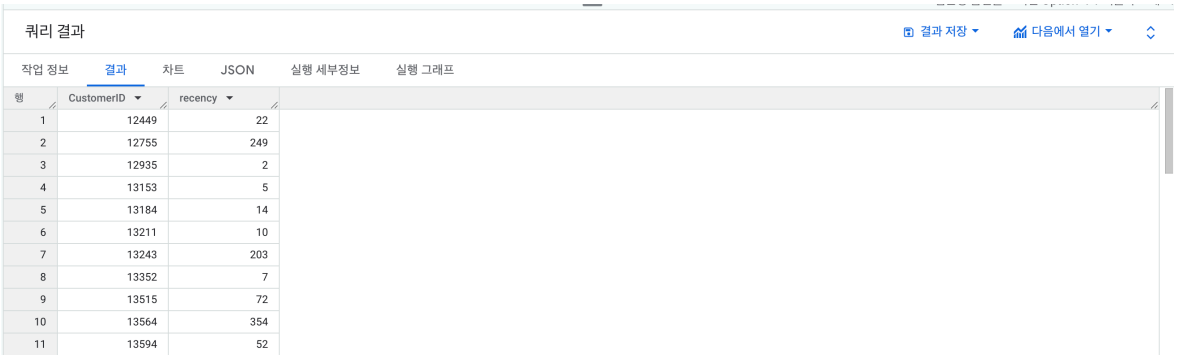
[결과 이미지를 넣어주세요]



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

```
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

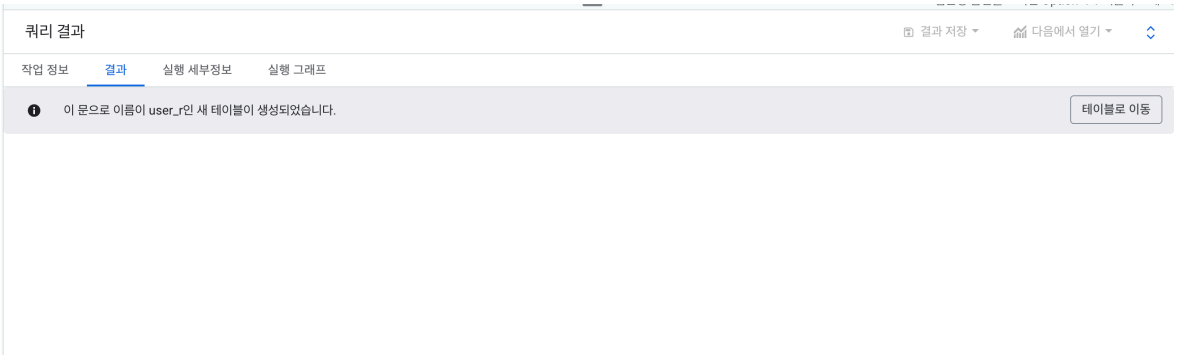


행	CustomerID	recency
1	12449	22
2	12755	249
3	12935	2
4	13153	5
5	13184	14
6	13211	10
7	13243	203
8	13352	7
9	13515	72
10	13564	354
11	13594	52

- **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.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

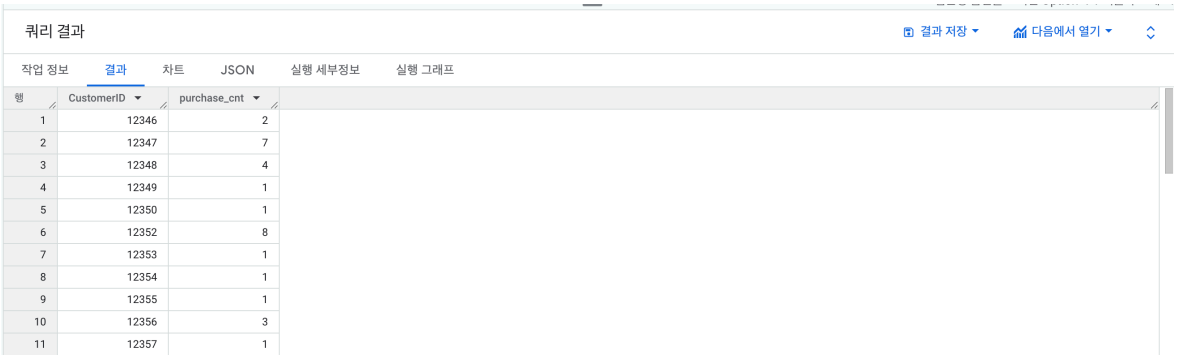


Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

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

[결과 이미지를 넣어주세요]

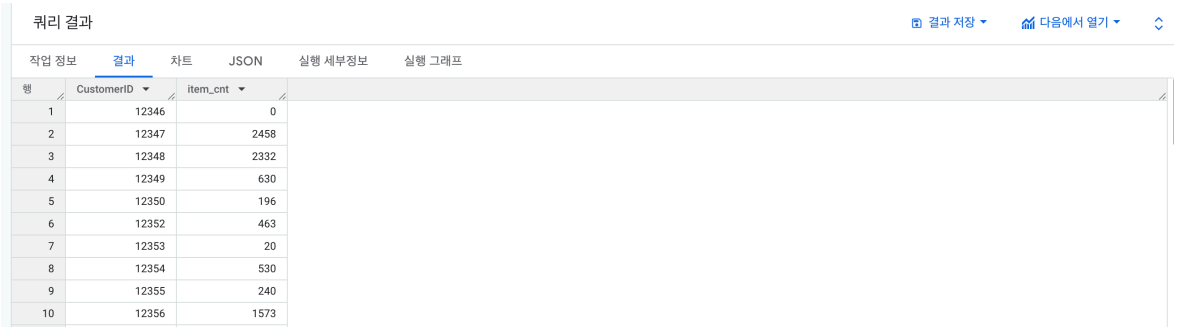


행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1
8	12354	1
9	12355	1
10	12356	3
11	12357	1

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

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

[결과 이미지를 넣어주세요]

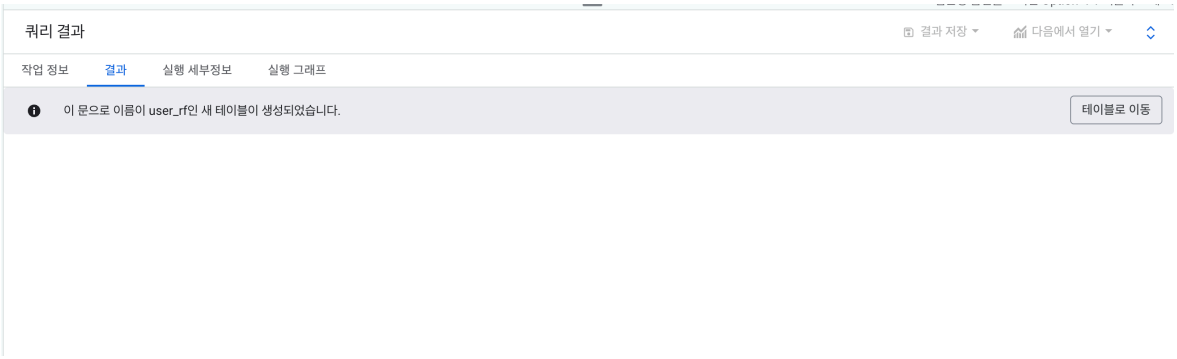


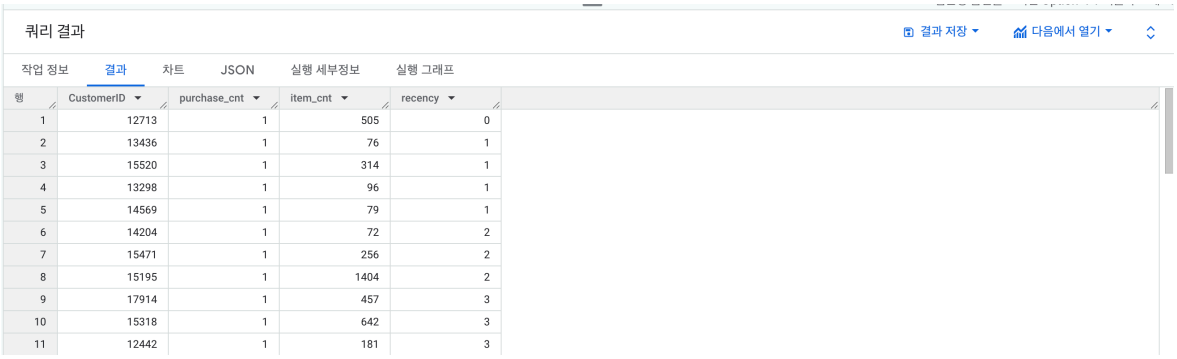
행	CustomerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240
10	12356	1573

- **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
WITH purchase_cnt AS (
 SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_cnt
 FROM modulabs_project.data
 GROUP BY CustomerID
),
item_cnt AS (
  SELECT CustomerID, SUM(Quantity) AS item_cnt
  FROM  modulabs_project.data
  GROUP BY CustomerID
)
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;
```

[결과 이미지를 넣어주세요]





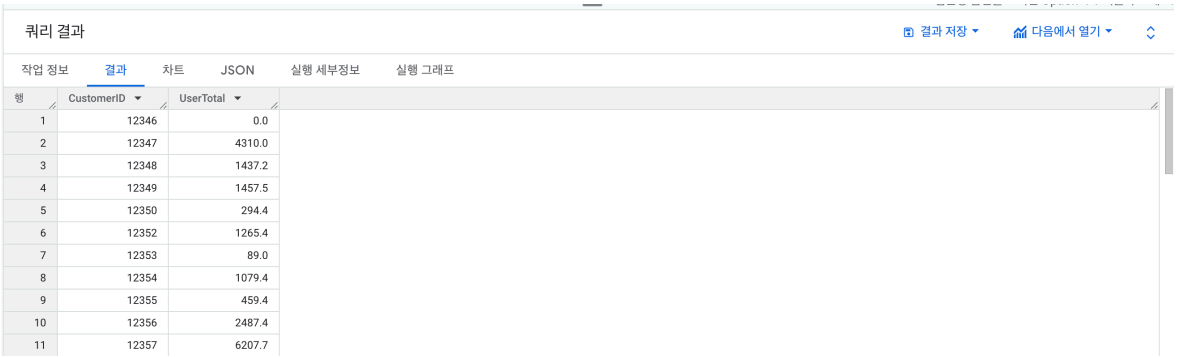
행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13436	1	76	1
3	15520	1	314	1
4	13298	1	96	1
5	14569	1	79	1
6	14204	1	72	2
7	15471	1	256	2
8	15195	1	1404	2
9	17914	1	457	3
10	15318	1	642	3
11	12442	1	181	3

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity*UnitPrice),1) AS UserTotal
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

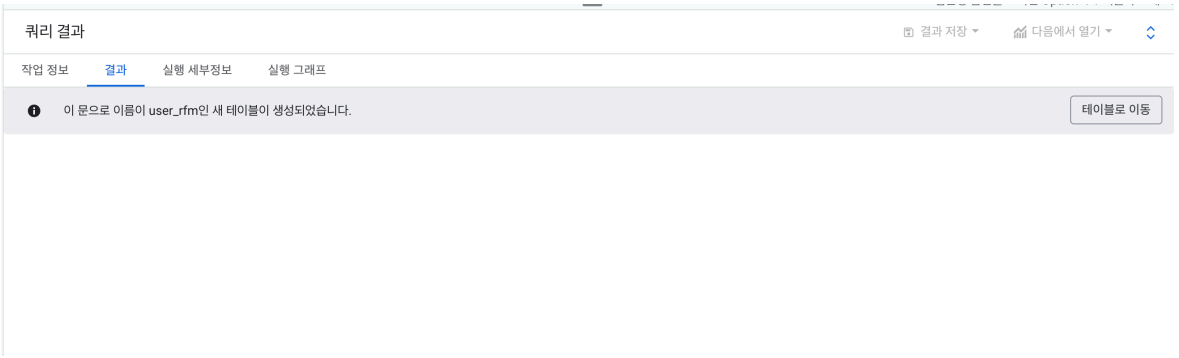


행	CustomerID	UserTotal
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7

- **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(user_total/purchase_cnt,0) AS user_average
FROM modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT CustomerID,
  ROUND(SUM(Quantity*UnitPrice),1) AS user_total
 FROM modulabs_project.data
 GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

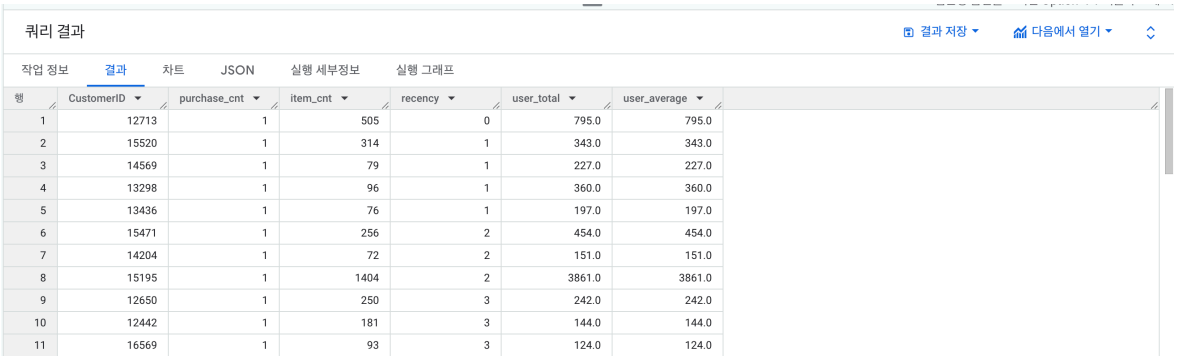


RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM 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	15520	1	314	1	343.0	343.0
3	14569	1	79	1	227.0	227.0
4	13298	1	96	1	360.0	360.0
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	12650	1	250	3	242.0	242.0
10	12442	1	181	3	144.0	144.0
11	16569	1	93	3	124.0	124.0

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]

2. 평균 구매 주기

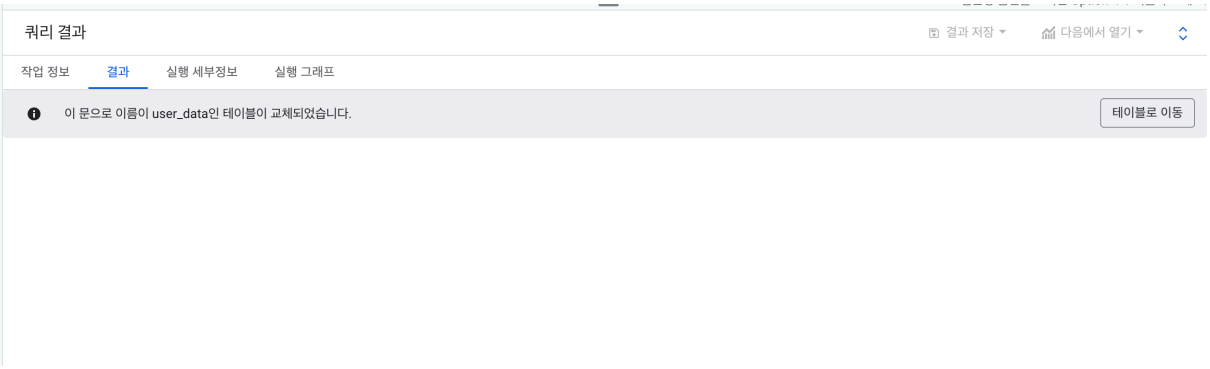
- **고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)**
  - **균 구매 소요 일수를 계산하고, 그 결과를** `user_data` **에 통합**

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

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_data AS u
```

```
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]



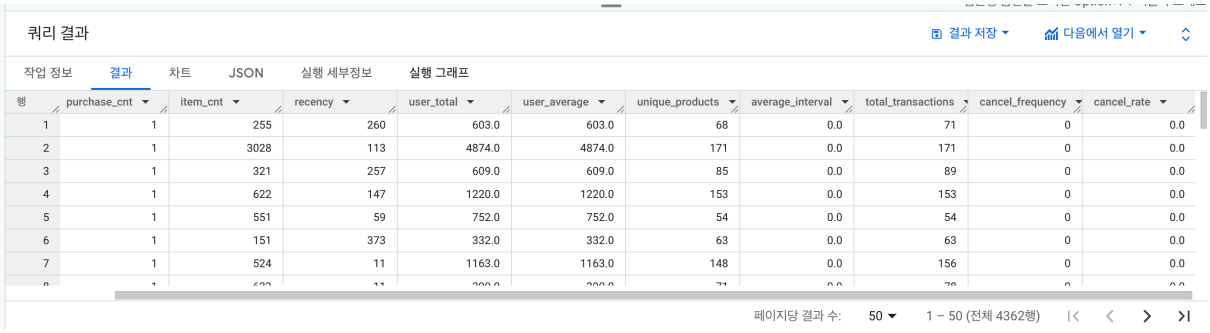
3. 구매 취소 경향성

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

```
CREATE OR REPLACE TABLE modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 END) AS cancel_frequency
  FROM modulabs_project.data
  GROUP BY CustomerID
)
SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate
FROM modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON t.CustomerID = u.CustomerID;
```

[결과 이미지를 넣어주세요]

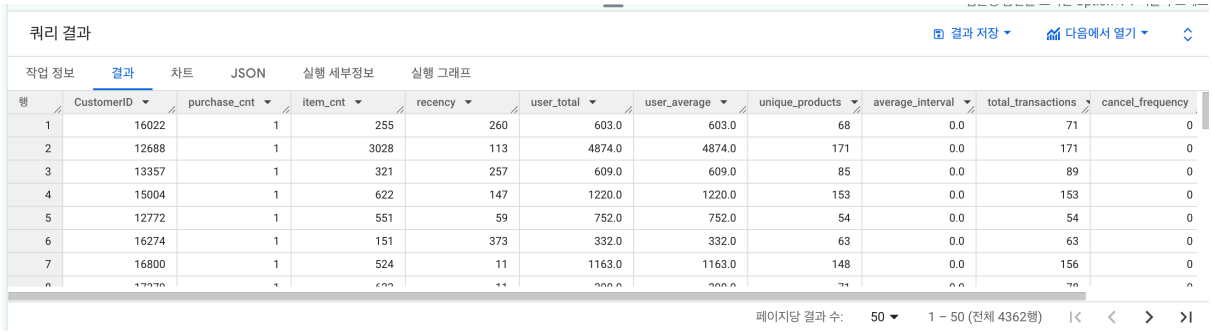


행	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	1	255	260	603.0	603.0	68	0.0	71	0	0.0
2	1	3028	113	4874.0	4874.0	171	0.0	171	0	0.0
3	1	321	257	609.0	609.0	85	0.0	89	0	0.0
4	1	622	147	1220.0	1220.0	153	0.0	153	0	0.0
5	1	551	59	752.0	752.0	54	0.0	54	0	0.0
6	1	151	373	332.0	332.0	63	0.0	63	0	0.0
7	1	524	11	1163.0	1163.0	148	0.0	156	0	0.0

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

```
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	total_transactions	cancel_frequency
1	16022	1	255	260	603.0	603.0	68	0.0	71	0
2	12688	1	3028	113	4874.0	4874.0	171	0.0	171	0
3	13357	1	321	257	609.0	609.0	85	0.0	89	0
4	15004	1	622	147	1220.0	1220.0	153	0.0	153	0
5	12772	1	551	59	752.0	752.0	54	0.0	54	0
6	16274	1	151	373	332.0	332.0	63	0.0	63	0
7	16800	1	524	11	1163.0	1163.0	148	0.0	156	0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

회고

[회고 내용을 작성해주세요]

Keep : 배운 내용을 토대로 프로젝트를 차근차근 진행하였다

Problem : 아직 숙련도가 부족해 중간중간 막혔다.

Try : 이런 프로젝트들을 통해 더욱 역량을 쌓아야겠다.