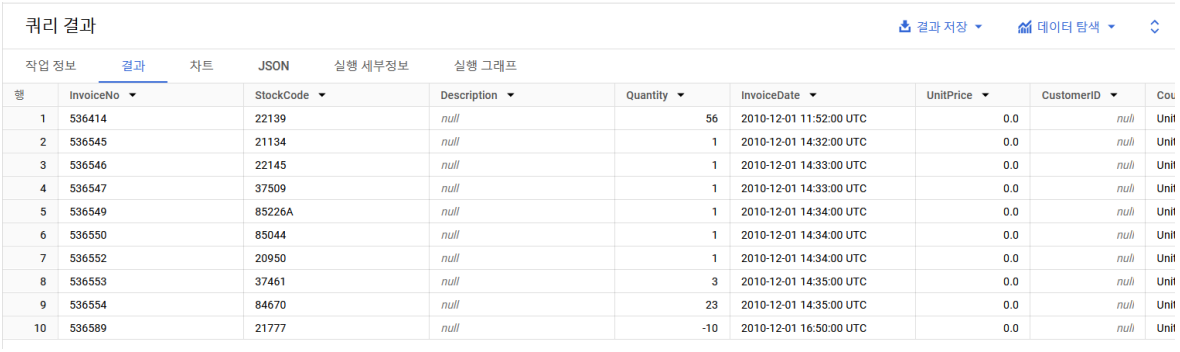
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM main-q-426001.modulabs_project.data
LIMIT 10
```



쿼리 결과

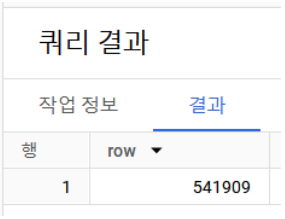
결과 저장 ▾ 데이터 탐색 ▾

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

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	null	Unit
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	Unit
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	Unit
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	null	Unit
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	null	Unit
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	null	Unit
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	0.0	null	Unit
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	null	Unit
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	0.0	null	Unit
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	0.0	null	Unit

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(InvoiceNo)
FROM main-q-426001.modulabs_project.data
```



쿼리 결과

작업 정보 | 결과

행	row
1	541909

총 541,909 개의 행을 가진 것을 확인

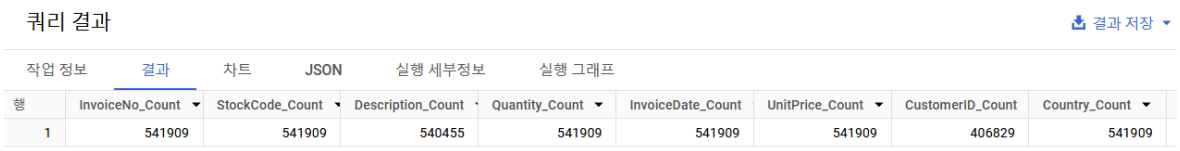
InvoiceNo	**각각의 고유한 거래를 나타내는 코드.이 코드가 'C'라는 글자로 시작한다면, 취소를 나타냅니다.하나의 거래에 여러 개의 제품이 함께 구매되었다면, 1개의 InvoiceNo에는 여러 개의 StockCode가 연결되어 있습니다.**
StockCode	**각각의 제품에 할당된 고유 코드**
Description	**각 제품에 대한 설명**
Quantity	**거래에서 제품을 몇 개 구매했는지에 대한 단위 수**
InvoiceDate	**거래가 일어난 날짜와 시간**
UnitPrice	**제품 당 단위 가격(영국 파운드)**
CustomerID	**각 고객에게 할당된 고유 식별자 코드**
Country	**주문이 발생한 국가**

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 main-q-426001.modulabs_project.data
```



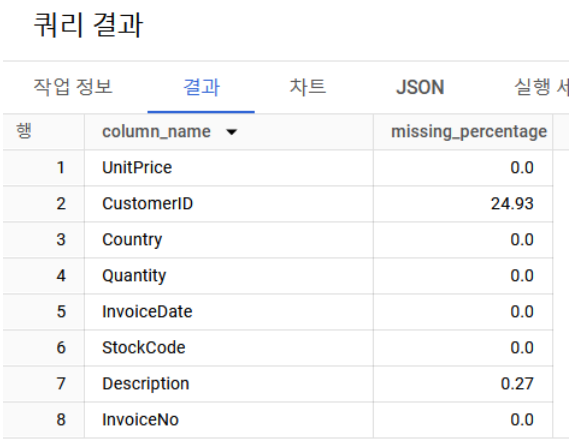
행	InvoiceNo_Count	StockCode_Count	Description_Count	Quantity_Count	InvoiceDate_Count	UnitPrice_Count	CustomerID_Count	Country_Count
1	541909	541909	540455	541909	541909	541909	406829	541909

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_percent
FROM main-q-426001.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 main-q-426001.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 main-q-426001.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 main-q-426001.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 main-q-426001.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 main-q-426001.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 main-q-426001.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 main-q-426001.modulabs_project.data
```

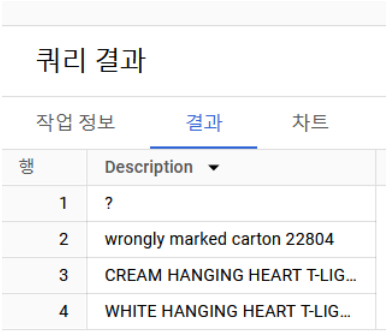


행	column_name	missing_percentage
1	UnitPrice	0.0
2	CustomerID	24.93
3	Country	0.0
4	Quantity	0.0
5	InvoiceDate	0.0
6	StockCode	0.0
7	Description	0.27
8	InvoiceNo	0.0

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM main-q-426001.modulabs_project.data
WHERE StockCode = '85123A'
```

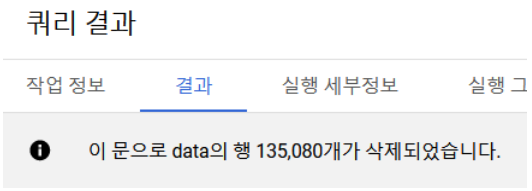


행	Description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	WHITE HANGING HEART T-LIG...

결측치 처리

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

```
DELETE FROM main-q-426001.modulabs_project.data
WHERE CustomerID IS NULL or Description = '?'
```



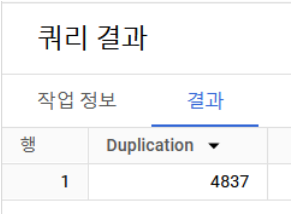
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT COUNT(*) AS Duplication
FROM(
SELECT COUNT(InvoiceNo) AS InvoiceNo_Count,
```

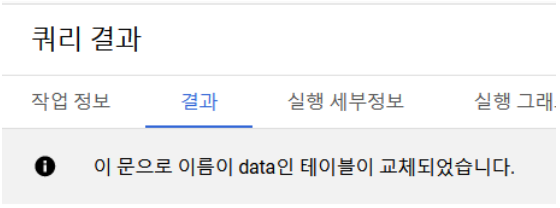
```
COUNT(StockCode) AS StockCode_Count,
COUNT(Description) AS Description_Count,
COUNT(Quantity) AS Quantity_Count,
COUNT(InvoiceDate) AS InvoiceDate_Count,
COUNT(UnitPrice) AS UnitPrice_Count,
COUNT(CustomerID) AS CustomerID_Count,
COUNT(Country) AS Country_Count
FROM main-q-426001.modulabs_project.data
GROUP BY InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
HAVING InvoiceNo_Count > 1
)
```



중복값 처리

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

```
## 중복값 제거
CREATE OR REPLACE TABLE main-q-426001.modulabs_project.data AS
SELECT DISTINCT *
FROM main-q-426001.modulabs_project.data;

## 처리 후 확인
SELECT COUNT(*)
FROM main-q-426001.modulabs_project.data
```

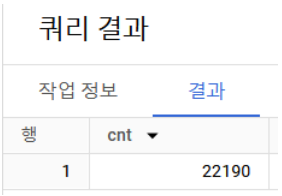


11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

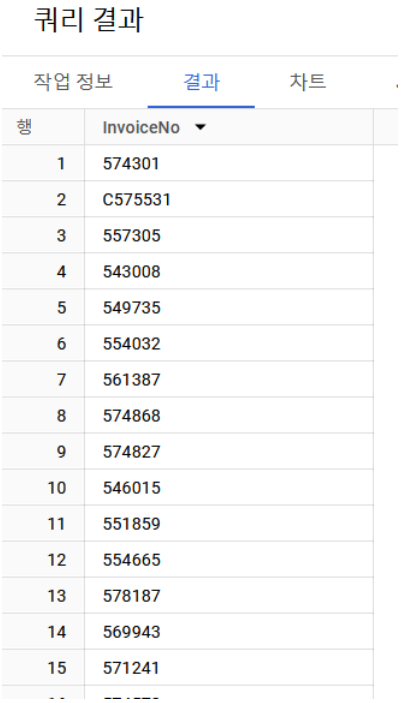
- **고유(unique)한 `InvoiceNo`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS cnt
FROM main-q-426001.modulabs_project.data
```



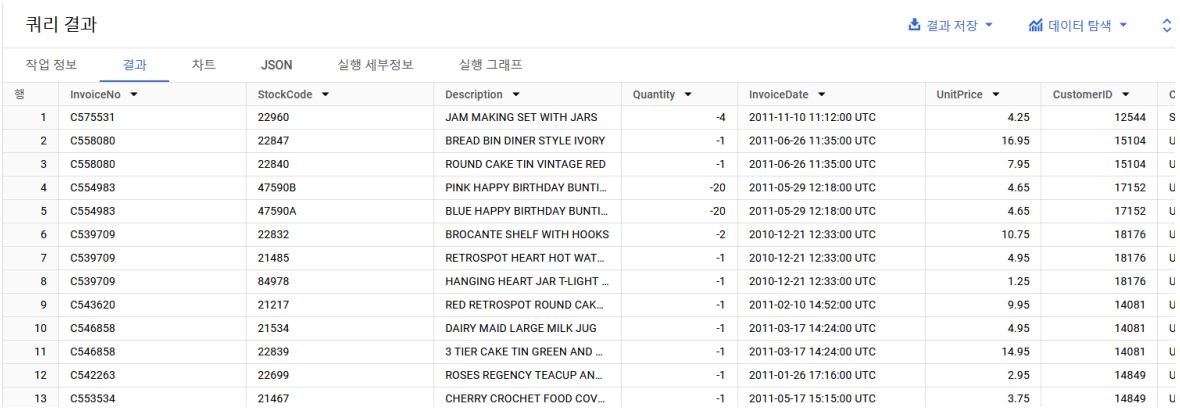
- **고유한 `InvoiceNo`를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM main-q-426001.modulabs_project.data
LIMIT 100
```



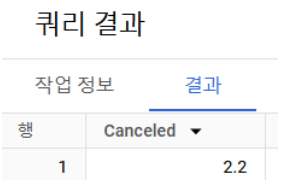
- **`InvoiceNo`가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)**

```
SELECT *
FROM main-q-426001.modulabs_project.data
WHERE InvoiceNO LIKE 'C%'
LIMIT 100
```



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

```
SELECT ROUND(SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END)/ COUNT(*) * 100, 1) AS Canceled
FROM main-q-426001.modulabs_project.data;
```



`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS cnt
FROM main-q-426001.modulabs_project.data
```

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM main-q-426001.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

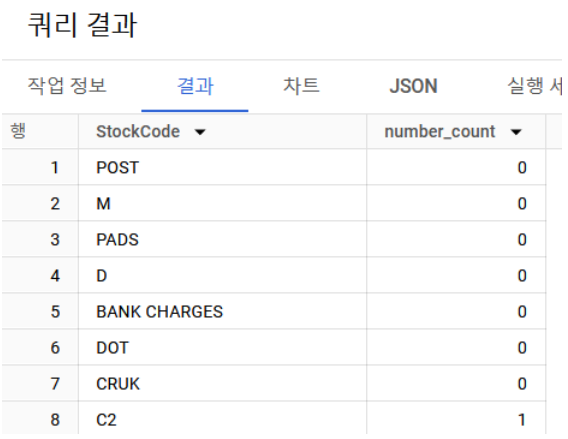
쿼리 결과

작업 정보 | 결과 | 차트 | JSON | 실행

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 main-q-426001.modulabs_project.data
)
WHERE number_count < 2
```



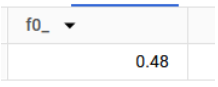
쿼리 결과

작업 정보 | 결과 | 차트 | JSON | 실행 서

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 ROUND(number_count / StockCode_cnt * 100, 2)
FROM(
  SELECT
  ( SELECT COUNT(StockCode)
    FROM main-q-426001.modulabs_project.data) AS StockCode_cnt,
    (
    SELECT COUNT(number_count)
    FROM (
      SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
      FROM main-q-426001.modulabs_project.data
      WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) < 2
    )
  ) AS number_count
)
```

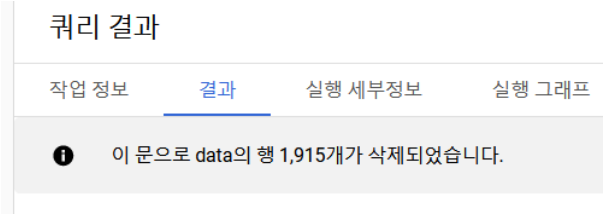


f0_
0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM main-q-426001.modulabs_project.data
WHERE StockCode IN (
```

```
SELECT DISTINCT StockCode
FROM (
  SELECT StockCode,
  FROM main-q-426001.modulabs_project.data
  WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) < 2
)
```



`Description` 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM main-q-426001.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30
```

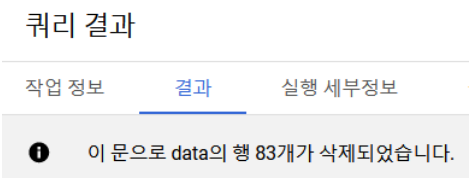
쿼리 결과

작업 정보 | 결과 | 차트 | JSON | 실행 세부

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
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000

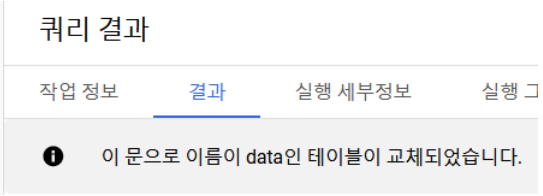
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE FROM main-q-426001.modulabs_project.data
WHERE Description LIKE 'Next%' OR Description LIKE 'High%'
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE main-q-426001.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM main-q-426001.modulabs_project.data
```



`UnitPrice` 살펴보기

- `UnitPrice`**의 최솟값, 최댓값, 평균을 구하기**

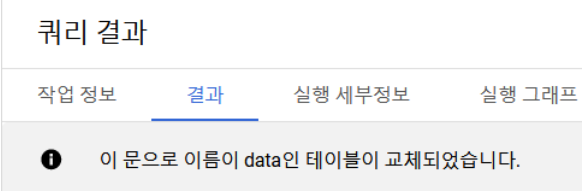
```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM main-q-426001.modulabs_project.data
```

- **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
FROM main-q-426001.modulabs_project.data
WHERE UnitPrice = 0.
```

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

```
CREATE OR REPLACE TABLE main-q-426001.modulabs_project.data AS
SELECT *
FROM main-q-426001.modulabs_project.data
WHERE UnitPrice != 0.0
```

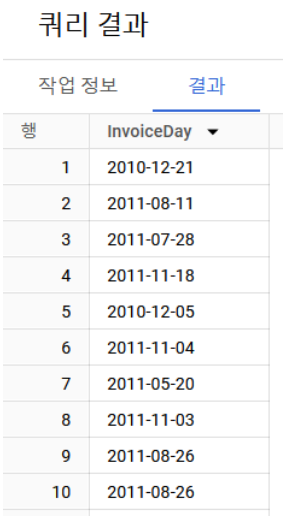


11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
SELECT CAST(InvoiceDate AS DATE) AS InvoiceDay
FROM main-q-426001.modulabs_project.data
```



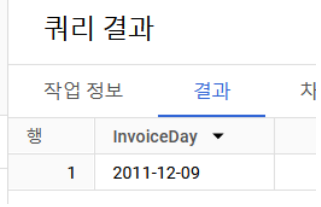
쿼리 결과

작업 정보 결과

행	InvoiceDay
1	2010-12-21
2	2011-08-11
3	2011-07-28
4	2011-11-18
5	2010-12-05
6	2011-11-04
7	2011-05-20
8	2011-11-03
9	2011-08-26
10	2011-08-26

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT CAST(InvoiceDate AS DATE) AS InvoiceDay
FROM main-q-426001.modulabs_project.data
ORDER BY InvoiceDay DESC
LIMIT 1
```



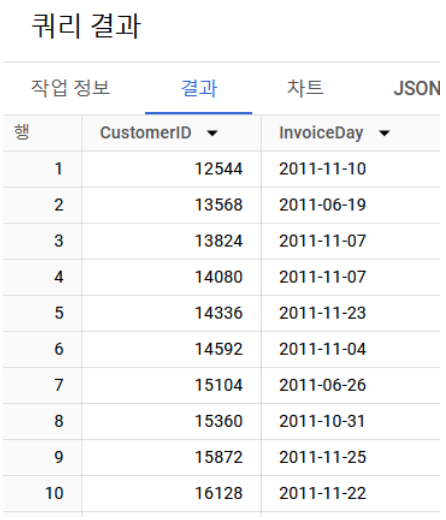
쿼리 결과

작업 정보 결과 치

행	InvoiceDay
1	2011-12-09

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

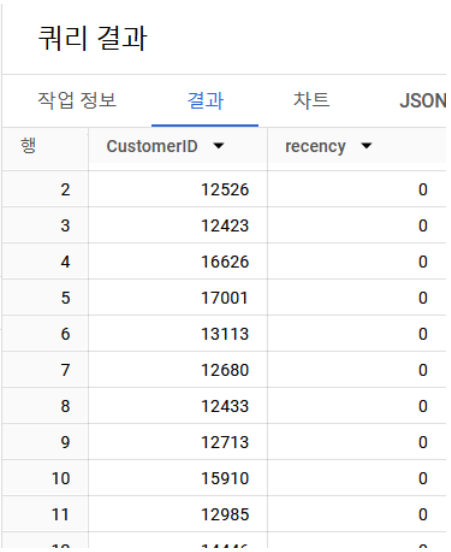
```
SELECT CustomerID,
MAX(CAST(InvoiceDate AS DATE)) AS InvoiceDay
FROM main-q-426001.modulabs_project.data
GROUP BY CustomerID
```



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
10	16128	2011-11-22

- **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
);
```

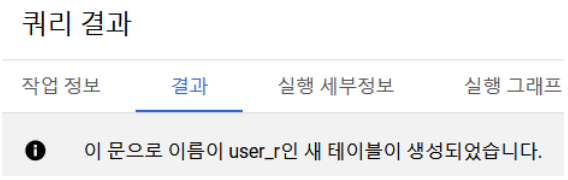


행	CustomerID	recency
2	12526	0
3	12423	0
4	16626	0
5	17001	0
6	13113	0
7	12680	0
8	12433	0
9	12713	0
10	15910	0
11	12985	0

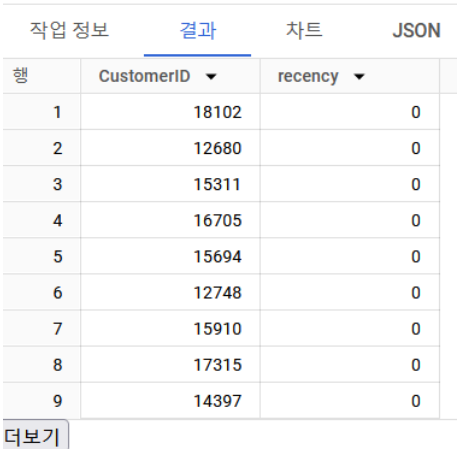
- **최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 `user_r`이라는 이름의 테이블로 저장하기**

```
CREATE OR REPLACE TABLE main-q-426001.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 main-q-426001.modulabs_project.data
  GROUP BY CustomerID
)
```




```
# user_r 테이블 확인

SELECT *
FROM main-q-426001.modulabs_project.user_r
```

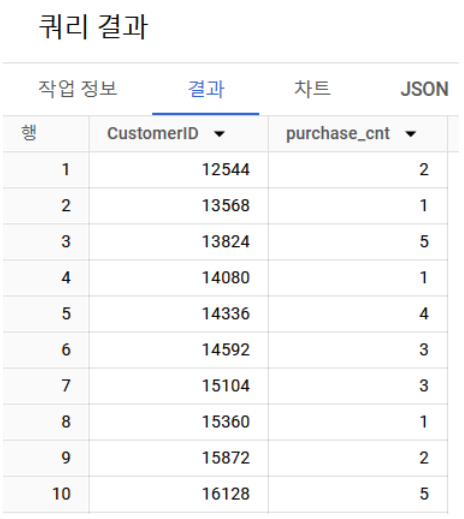


행	CustomerID	recency
1	18102	0
2	12680	0
3	15311	0
4	16705	0
5	15694	0
6	12748	0
7	15910	0
8	17315	0
9	14397	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

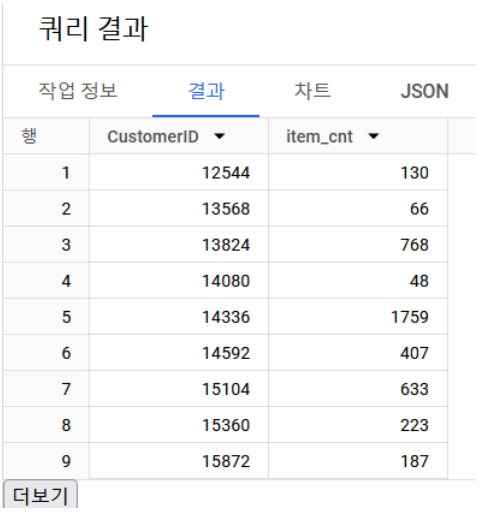
```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM main-q-426001.modulabs_project.data
GROUP BY CustomerID
```



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
10	16128	5

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM main-q-426001.modulabs_project.data
GROUP BY CustomerID
```



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
9	15872	187

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

```
CREATE OR REPLACE TABLE main-q-426001.modulabs_project.user_rf AS
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM main-q-426001.modulabs_project.data
  GROUP BY CustomerID
),
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
```

```
  FROM main-q-426001.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 main-q-426001.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

쿼리 결과

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

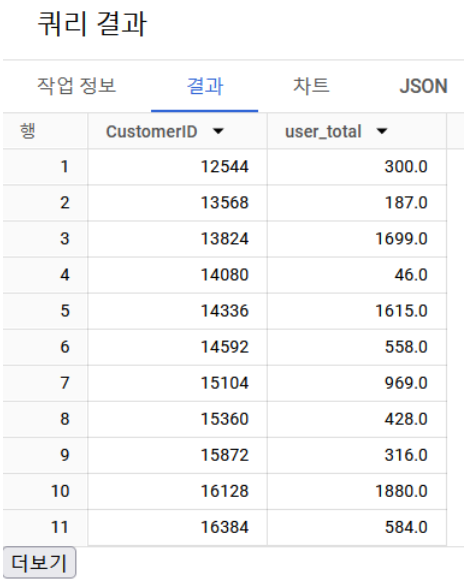
행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	18010	1	60	256
3	15083	1	38	256
4	12792	1	215	256
5	15520	1	314	1
6	14569	1	79	1
7	13298	1	96	1
8	13436	1	76	1
9	13357	1	321	257

더보기

Monetary

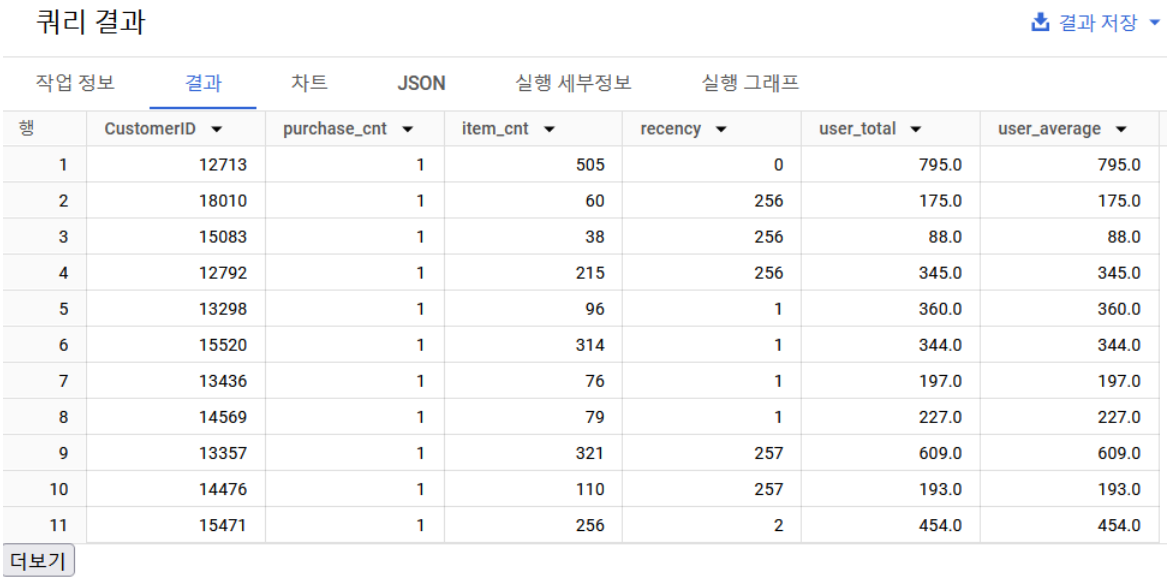
- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity),0) AS user_total
FROM main-q-426001.modulabs_project.data
GROUP BY CustomerID
```



- **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 main-q-426001.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 main-q-426001.modulabs_project.user_rf rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity),0) AS user_total
  FROM main-q-426001.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```



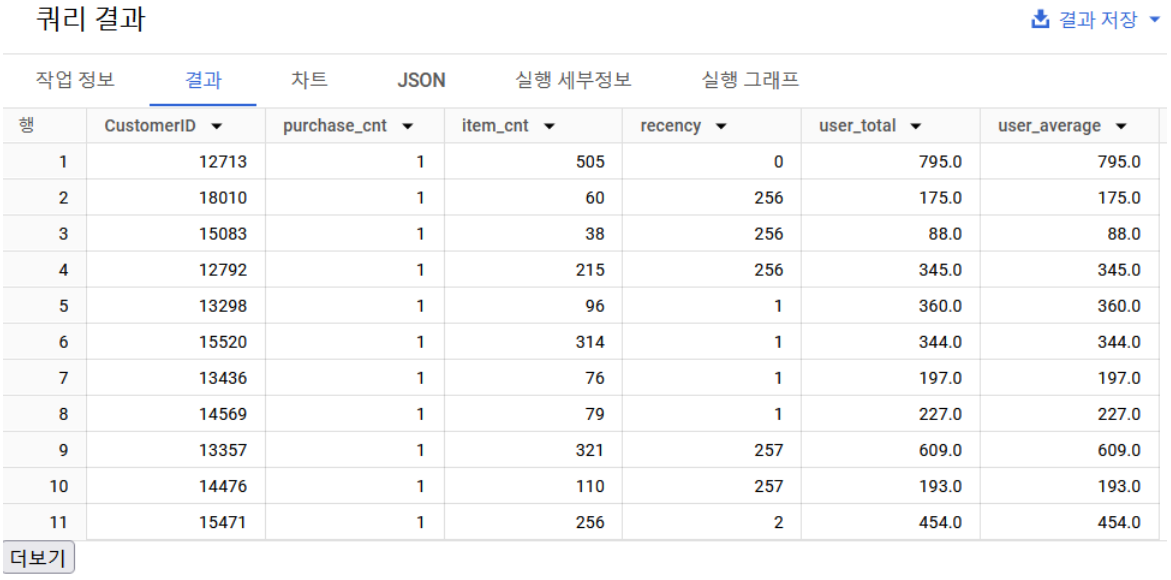
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	18010	1	60	256	175.0	175.0
3	15083	1	38	256	88.0	88.0
4	12792	1	215	256	345.0	345.0
5	13298	1	96	1	360.0	360.0
6	15520	1	314	1	344.0	344.0
7	13436	1	76	1	197.0	197.0
8	14569	1	79	1	227.0	227.0
9	13357	1	321	257	609.0	609.0
10	14476	1	110	257	193.0	193.0
11	15471	1	256	2	454.0	454.0

더보기

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM main-q-426001.modulabs_project.user_rfm
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	18010	1	60	256	175.0	175.0
3	15083	1	38	256	88.0	88.0
4	12792	1	215	256	345.0	345.0
5	13298	1	96	1	360.0	360.0
6	15520	1	314	1	344.0	344.0
7	13436	1	76	1	197.0	197.0
8	14569	1	79	1	227.0	227.0
9	13357	1	321	257	609.0	609.0
10	14476	1	110	257	193.0	193.0
11	15471	1	256	2	454.0	454.0

더보기

11-8. 추가 Feature 추출

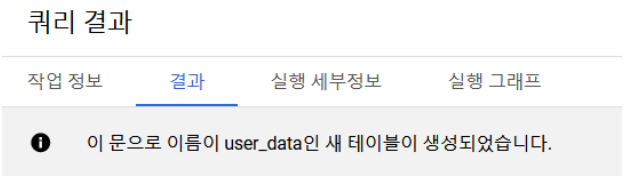
1. 구매하는 제품의 다양성

- **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;
```




```
SELECT *
FROM main-q-426001.modulabs_project.user_data
```

쿼리 결과 결과 저장 데이터 탐색

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

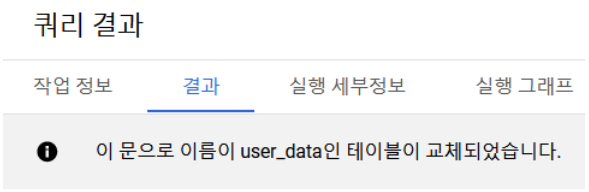
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15195	1	1404	2	3861.0	3861.0	1
2	15313	1	25	110	52.0	52.0	1
3	17102	1	2	261	26.0	26.0	1
4	15488	1	72	92	76.0	76.0	1
5	14576	1	12	372	35.0	35.0	1
6	16093	1	20	106	17.0	17.0	1
7	16579	1	-12	365	-31.0	-31.0	1
8	14424	1	48	17	322.0	322.0	1
9	16881	1	600	66	432.0	432.0	1
10	17752	1	192	359	81.0	81.0	1
11	16995	1	-1	372	-1.0	-1.0	1

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
```

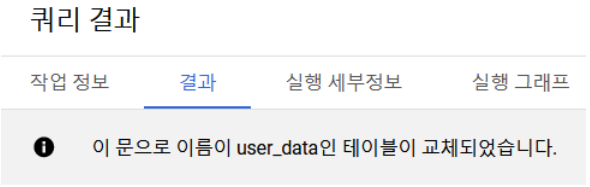
```
      project_name.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

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;
```



3. 구매 취소 경향성

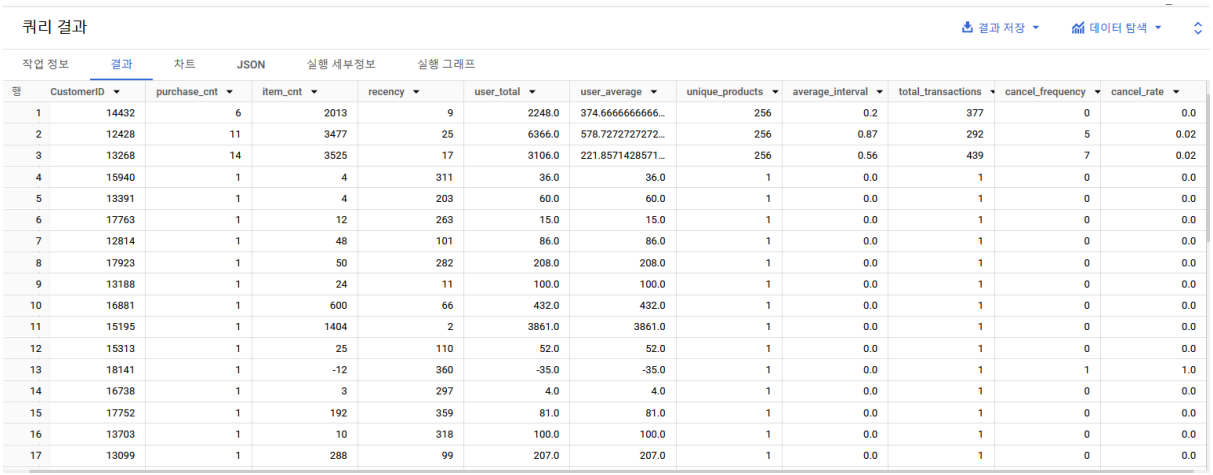
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE main-q-426001.modulabs_project.user_data AS
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(CustomerID) AS total_transactions,
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE NULL END) AS cancel_frequency
  FROM main-q-426001.modulabs_project.data
  GROUP BY CustomerID
)
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(cancel_frequency / total_transactions, 2) AS cancel_rate
FROM main-q-426001.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 main-q-426001.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	cancel_frequency	cancel_rate
1	14432	6	2013	9	2248.0	374.6666666666...	256	0.2	377	0	0.0
2	12428	11	3477	25	6366.0	578.7272727272...	256	0.87	292	5	0.02
3	13268	14	3525	17	3106.0	221.8571428571...	256	0.56	439	7	0.02
4	15940	1	4	311	36.0	36.0	1	0.0	1	0	0.0
5	13391	1	4	203	60.0	60.0	1	0.0	1	0	0.0
6	17763	1	12	263	15.0	15.0	1	0.0	1	0	0.0
7	12814	1	48	101	86.0	86.0	1	0.0	1	0	0.0
8	17923	1	50	282	208.0	208.0	1	0.0	1	0	0.0
9	13188	1	24	11	100.0	100.0	1	0.0	1	0	0.0
10	16881	1	600	66	432.0	432.0	1	0.0	1	0	0.0
11	15195	1	1404	2	3861.0	3861.0	1	0.0	1	0	0.0
12	15313	1	25	110	52.0	52.0	1	0.0	1	0	0.0
13	18141	1	-12	360	-35.0	-35.0	1	0.0	1	1	1.0
14	16738	1	3	297	4.0	4.0	1	0.0	1	0	0.0
15	17752	1	192	359	81.0	81.0	1	0.0	1	0	0.0
16	13703	1	10	318	100.0	100.0	1	0.0	1	0	0.0
17	13099	1	288	99	207.0	207.0	1	0.0	1	0	0.0

user_data.csv