

# 고객을 세그먼테이션하자 [프로젝트]

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.data
LIMIT 10
```

[결과 이미지를 넣어주세요]

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*) AS TOTAL_ROW_COUNT
FROM bright-meridian-439401-g6.modulabs_project1.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	TOTAL_ROW_COUNT
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 bright-meridian-439401-g6.modulabs_project1.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프
행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	CO
1	541909	541909	540455	541909	541909	

페이지당 결과 수: 50

1 - 1 (전체 1행)

작업 기록

새로고침

## 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 ALL**을 통해 합치기

```
SELECT column_name, ROUND((total - column_value) / total * 100, 2) AS missing_percentage
FROM
(
    SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM brig
    SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM brig
    SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM
    SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM bright
    SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM
    SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM brig
    SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM br
    SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM bright-m
) AS column_data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

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

## 결측치 처리 전략

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

```
SELECT Description
FROM bright-meridian-439401-g6.modulabs_project1.data
WHERE StockCode = '85123A'
LIMIT 4
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

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

## 결측치 처리

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

```
DELETE FROM bright-meridian-439401-g6.modulabs_project1.data2
WHERE Description IS NULL
OR CustomerID IS NULL
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 실행 세부정보 실행 그래프

**i** 이 문으로 data2의 행 135,080개가 삭제되었습니다.

테이블로 이동

## 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_rows
FROM (
    SELECT InvoiceNo , StockCode , Description , Quantity , InvoiceDate , UnitPrice, CustomerID , Coun
    FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY InvoiceNo , StockCode , Description , Quantity , InvoiceDate , UnitPrice, CustomerID , Coun
HAVING COUNT(*) > 1)
```

[결과 이미지를 넣어주세요]

The screenshot shows a web interface for query results. At the top, it says '쿼리 결과' (Query Result). Below that, there are tabs: '작업 정보' (Job Info), '결과' (Result), '차트' (Chart), 'JSON', '실행 세부정보' (Execution Details), and '실행 그래프' (Execution Graph). The '결과' tab is selected. It shows a table with two columns: '행' (Row) and 'duplicate\_rows'. The first row shows '1' and '4837'.

행	duplicate_rows
1	4837

## 중복값 처리

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

```
CREATE OR REPLACE TABLE `bright-meridian-439401-g6.modulabs_project1.data1` AS(
SELECT DISTINCT *
FROM bright-meridian-439401-g6.modulabs_project1.data1)
```

[결과 이미지를 넣어주세요]

The screenshot shows a web interface for query results. At the top, it says '쿼리 결과' (Query Result). Below that, there are tabs: '작업 정보' (Job Info), '결과' (Result), '실행 세부정보' (Execution Details), and '실행 그래프' (Execution Graph). The '결과' tab is selected. It shows a message: '이 문으로 이름이 data1인 테이블이 교체되었습니다.' (The table named data1 is replaced by this statement.) and a button '테이블로 이동' (Go to Table).

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 data1인 테이블이 교체되었습니다.			

## 11-6. 데이터 전처리(3): 오류값 처리

**InvoiceNo** 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo) AS Unique_InvoiceNo_count
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

쿼리 결과		결과 저장	데이터 탐색
작업 정보	결과	차트	JSON
실행 세부정보	실행 그래프		
행	Unique_InvoiceNo_count		
1	22190		

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM bright-meridian-439401-g6.modulabs_project1.data1
LIMIT 100
```

[결과 이미지를 넣어주세요]

쿼리 결과		결과 저장	데이터 탐색
작업 정보	결과	차트	JSON
실행 세부정보	실행 그래프		
행	InvoiceNo		
1	574301		
2	C575531		
3	557305		
4	543008		
5	549735		
6	554032		
7	561387		
8	574868		
9	574827		
10	546015		
11	551859		
12	554665		
13	578187		
14	569943		
15	571241		
16	574573		

페이지당 결과 수: 50 1 - 50 (전체 100행) |< < > >|

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

```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity
1	C575531	22960	JAM MAKING SET WITH JARS	
2	C558080	22840	ROUND CAKE TIN VINTAGE RED	
3	C558080	22847	BREAD BIN DINER STYLE IVORY	
4	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	
5	C554983	47590B	PINK HAPPY BIRTHDAY BUNTI...	
6	C539709	84978	HANGING HEART JAR T-LIGHT ...	
7	C539709	22832	BROCANTE SHELF WITH HOOKS	
8	C539709	21485	RETROSPOT HEART HOT WAT...	
9	C543620	21217	RED RETROSPOT ROUND CAK...	
10	C546858	22839	3 TIER CAKE TIN GREEN AND ...	
11	C546858	21534	DAIRY MAID LARGE MILK JUG	
12	C542263	22699	ROSES REGENCY TEACUP AN...	
13	C553534	21467	CHERRY CROCHET FOOD COV...	
14	C570996	22909	SET OF 20 VINTAGE CHRISTM...	
15	C570996	23376	PACK OF 12 VINTAGE CHRIST...	

- 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) AS Can
```

```
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	Canceled_Rate
1	2.2

페이지당 결과 수: 50

1 - 1 (전체 1행)

<<

<

>

>>

작업 기록

새로고침

## StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT stockcode) AS stockcode_count
```

```
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	데이터 탐색	
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	stockcode_count				
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 bright-meridian-439401-g6.modulabs_project1.data1
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				
9	22197	1110				
10	23203	1108				

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 값들에는 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 bright-meridian-439401-g6.modulabs_project1.data1
)
WHERE number_count < 2
```

[결과 이미지를 넣어주세요]

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)

- StockCode의 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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
SELECT ROUND(SUM(CASE WHEN number_count < 2 THEN 1 ELSE 0 END)/ COUNT(number_count) *100, 2) AS num
FROM (
    SELECT StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM bright-meridian-439401-g6.modulabs_project1.data1
)
```

[결과 이미지를 넣어주세요]



쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	number_count_Rate
1	0.48

페이지당 결과 수: 501 - 1 (전체 1행)

작업 기록

새로고침

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE StockCode IN (
    SELECT DISTINCT StockCode
    FROM (
        SELECT StockCode,
            LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
        FROM bright-meridian-439401-g6.modulabs_project1.data1 )
    WHERE number_count < 2
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

실행 세부정보

실행 그래프

이 문으로 data1의 행 1,915개가 삭제되었습니다.

테이블로 이동

작업 기록

새로고침

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description , COUNT(*) AS description_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY Description
LIMIT 30
```

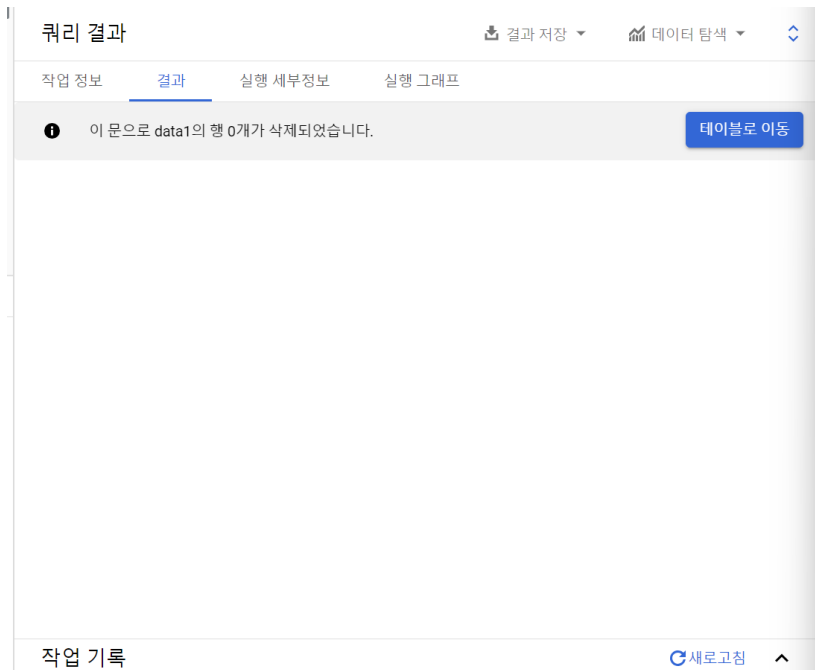
[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	데이터 탐색	
작업 정보			결과	차트	JSON
행	Description	description_cnt			
1	ASSORTED COLOUR MINI CAS...	372			
2	JAM MAKING SET WITH JARS	966			
3	PINK BLUE FELT CRAFT TRINK...	465			
4	SET OF 6 RIBBONS VINTAGE C...	343			
5	ASSORTED COLOUR BIRD ORN...	1405			
6	PAPER CHAIN KIT 50'S CHRIST...	1013			
7	EMBROIDERED RIBBON REEL R...	49			
8	TRADITIONAL KNITTING NANCY	523			
9	FELTCRAFT PRINCESS OLIVIA ...	254			
10	EMBROIDERED RIBBON REEL E...	87			
11	PAPER CHAIN KIT VINTAGE CH...	718			
12	CHRISTMAS CRAFT LITTLE FRI...	433			
13	SCANDINAVIAN REDS RIBBONS	343			
14	TRADITIONAL CHRISTMAS RIB...	272			
15	EMBROIDERED RIBBON REEL S...	63			
16	FELTCRAFT PRINCESS LOLA D...	383			
17	SET OF 4 KNICK KNACK TINS ...	328			
페이지당 결과 수: 50			1 - 30 (전체 30행)		
작업 기록			새로고침		

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE Description LIKE '%Next Day Carriage%'
AND Description LIKE "%High Resolution Image%"
```

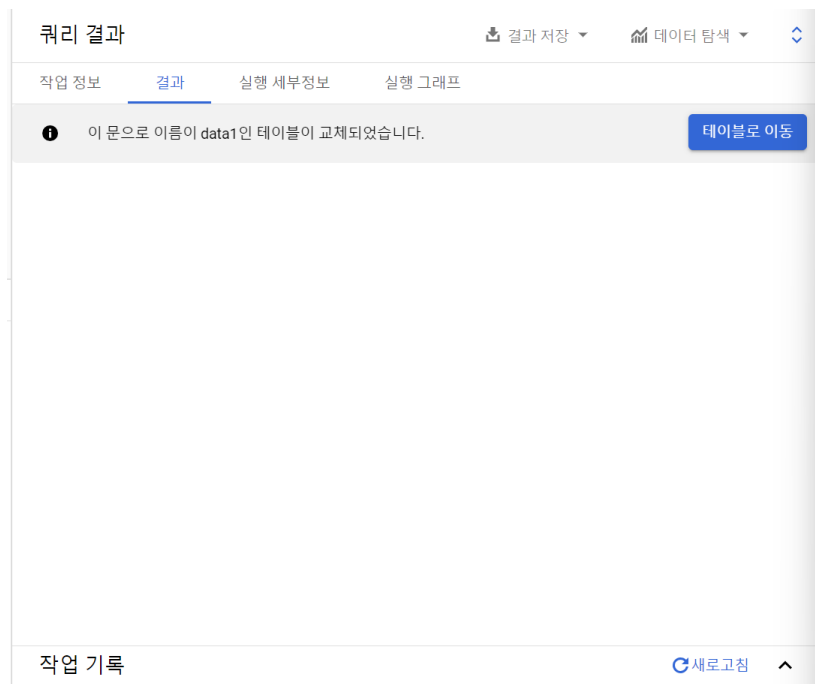
[결과 이미지를 넣어주세요]



- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.data1 AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]



**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 bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	min_price	max_price	avg_price	
1	0.0	649.5	2.907457172951...	

페이지당 결과 수: 50

1 - 1 (전체 1행)

<

<

>

>

작업 기록

새로고침

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

```
SELECT COUNTIF(quantity = 0) AS cnt_quantity, MIN(quantity) AS min_quantity, MAX(quantity) AS max_qua
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장   데이터 탐색

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

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	0	-80995	80995	12.22938334555...

페이지당 결과 수: 50   1 - 1 (전체 1행)   < >

작업 기록   새로고침

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.data1 AS
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.data1
WHERE Unitprice != 0
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장   데이터 탐색

작업 정보   **결과**   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 data1인 테이블이 교체되었습니다.   [테이블로 이동](#)

작업 기록   새로고침

## 11-7. RFM 스코어

## Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *  
FROM bright-meridian-439401-g6.modulabs_project1.data1
```

[결과 이미지를 넣어주세요]

쿼리 결과					
<a href="#">결과 저장</a> <a href="#">데이터 탐색</a>					
작업 정보 <b>결과</b> 차트   JSON   실행 세부정보   실행 그래프					
행	InvoiceDay	InvoiceNo	StockCode	Quantity	
1	2011-11-03	574301	23512	6	
2	2011-11-03	574301	84879	8	
3	2011-11-03	574301	22734	6	
4	2011-11-03	574301	22910	6	
5	2011-11-03	574301	22960	6	
6	2011-11-03	574301	23240	6	
7	2011-11-03	574301	85049E	12	
8	2011-11-03	574301	23514	6	
9	2011-11-03	574301	23511	6	
10	2011-11-03	574301	22750	4	
11	2011-11-03	574301	22144	6	
12	2011-11-03	574301	20971	12	
13	2011-11-03	574301	85049A	12	
14	2011-11-03	574301	20749	4	
15	2011-11-03	574301	22621	12	
16	2011-11-03	574301	22086	6	
페이지당 결과 수: 50   1 - 50 (전체 399656행)   < > >					
작업 기록 <a href="#">새로고침</a> ^					

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT  
    MAX(InvoiceDate) AS most_recent_date,  
    DATE(InvoiceDate) AS InvoiceDay  
FROM bright-meridian-439401-g6.modulabs_project1.data1  
GROUP BY DATE(InvoiceDate);
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 ▼ 데이터 탐색 ▼

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	most_recent_date ▼	InvoiceDay ▼
1	2011-11-03 19:50:00 UTC	2011-11-03
2	2011-11-10 20:01:00 UTC	2011-11-10
3	2011-06-19 16:11:00 UTC	2011-06-19
4	2011-02-02 17:15:00 UTC	2011-02-02
5	2011-04-12 17:12:00 UTC	2011-04-12
6	2011-05-20 16:56:00 UTC	2011-05-20
7	2011-07-27 17:08:00 UTC	2011-07-27
8	2011-11-07 17:50:00 UTC	2011-11-07
9	2011-03-08 17:28:00 UTC	2011-03-08
10	2011-05-04 17:19:00 UTC	2011-05-04
11	2011-05-25 17:31:00 UTC	2011-05-25
12	2011-11-23 17:45:00 UTC	2011-11-23
13	2011-10-06 20:38:00 UTC	2011-10-06
14	2011-10-14 17:23:00 UTC	2011-10-14
15	2011-11-04 18:11:00 UTC	2011-11-04
16	2011-03-02 17:33:00 UTC	2011-03-02
17	2011-05-27 17:14:00 UTC	2011-05-27

페이지당 결과 수: 50 ▼ 1 - 50 (전체 305행) |< < > >|

작업 기록 새로고침 ^

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
    CustomerID,
    MAX(InvoiceDate) AS InvoiceDay
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 ▼ 데이터 탐색 ▼

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID ▼	InvoiceDay ▼
1	12544	2011-11-10 11:12:00 UTC
2	13568	2011-06-19 14:42:00 UTC
3	13824	2011-11-07 12:41:00 UTC
4	14080	2011-11-07 11:09:00 UTC
5	14336	2011-11-23 11:40:00 UTC
6	14592	2011-11-04 16:35:00 UTC
7	15104	2011-06-26 11:35:00 UTC
8	15360	2011-10-31 09:35:00 UTC
9	15872	2011-11-25 11:55:00 UTC
10	16128	2011-11-22 11:14:00 UTC
11	16384	2011-09-11 11:19:00 UTC
12	17152	2011-05-29 12:19:00 UTC
13	17408	2011-06-29 12:53:00 UTC
14	17664	2011-11-21 11:29:00 UTC
15	17920	2011-12-05 14:29:00 UTC
16	18176	2010-12-21 12:33:00 UTC
17	12545	2011-09-25 13:39:00 UTC

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행) |< < > >|

작업 기록 새로고침 ^

- 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 bright-meridian-439401-g6.modulabs_project1.data1
  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	13568	173
2	17154	75
3	15117	14
4	14870	261
5	15894	253
6	14362	22
7	12573	227
8	15133	127
9	17451	1
10	15410	84
11	16189	15
12	16193	236
13	16198	75
14	16711	3
15	17223	310
16	14408	10
17	14410	15

페이지당 결과 수: 50 1 - 50 (전체 4362행)

작업 기록 새로고침

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

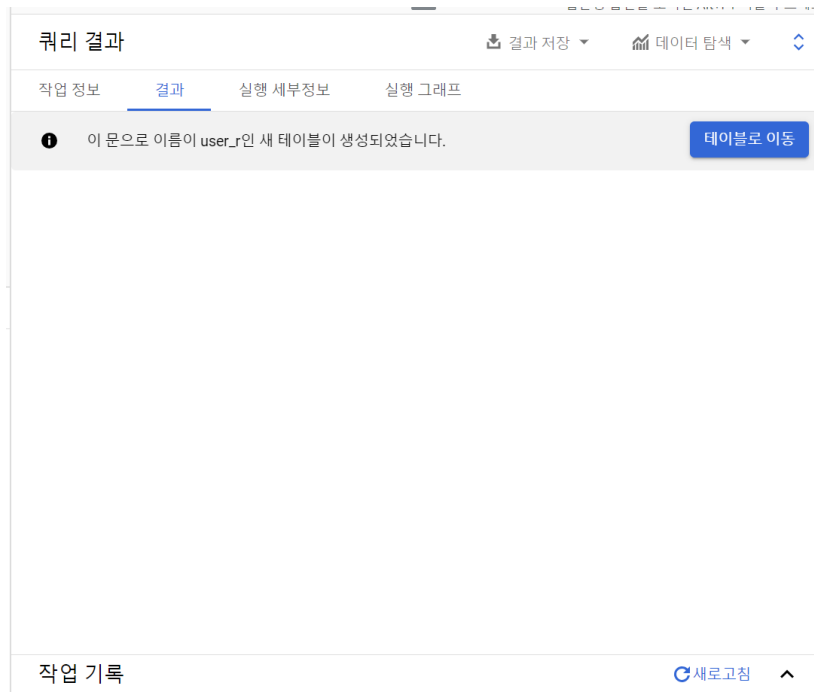
```

CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.user_r AS
(SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM bright-meridian-439401-g6.modulabs_project1.data1
  GROUP BY CustomerID
))

```

[결과 이미지를 넣어주세요]





## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

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
11	16384	2
12	17152	4
13	17408	1
14	17664	2
15	17920	17
16	18176	2
17	12545	2

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
FROM bright-meridian-439401-g6.modulabs_project1.data1
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			
9	15872	187			
10	16128	988			
11	16384	260			
12	17152	477			
13	17408	3			
14	17664	604			
15	17920	2471			
16	18176	279			
17	12545	517			

페이지당 결과 수: 50 1 - 50 (전체 4362행) |< < > >|

작업 기록 새로고침

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

```
CREATE OR REPLACE TABLE bright-meridian-439401-g6.modulabs_project1.user_rf AS

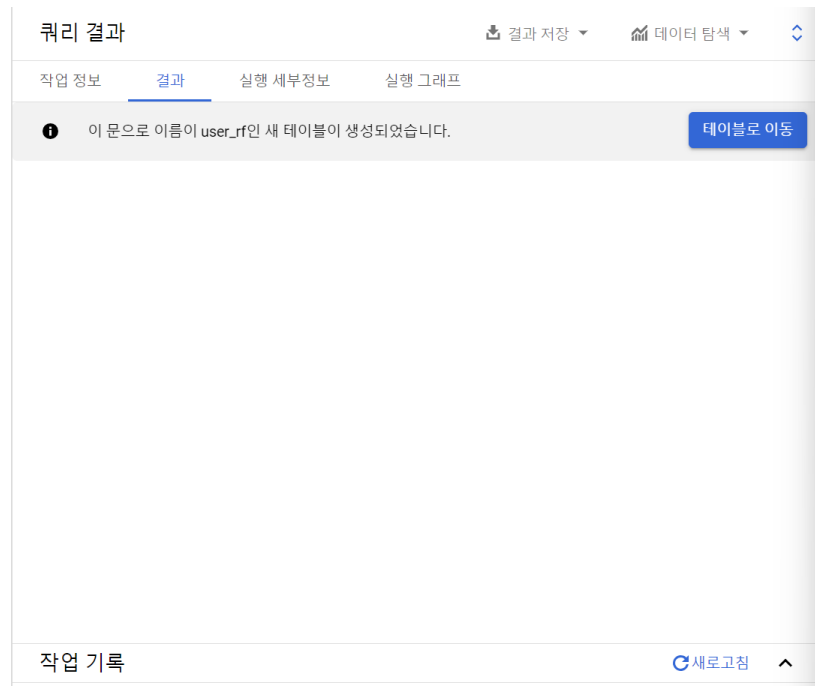
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT InvoiceNo) AS purchase_cnt
    FROM bright-meridian-439401-g6.modulabs_project1.data1
    GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
    SELECT
        CustomerID,
        SUM(Quantity) AS item_cnt
    FROM bright-meridian-439401-g6.modulabs_project1.data1
    GROUP BY CustomerID
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
    pc.CustomerID,
    pc.purchase_cnt,
    ic.item_cnt,
    ur.reccency
FROM purchase_cnt AS pc
```

```
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN bright-meridian-439401-g6.modulabs_project1.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),0) AS user_total
FROM bright-meridian-439401-g6.modulabs_project1.data1
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	54.0
2	13568	131.0
3	13824	127.0
4	14080	4.0
5	14336	145.0
6	14592	324.0
7	15104	365.0
8	15360	31.0
9	15872	226.0
10	16128	216.0
11	16384	116.0
12	17152	58.0
13	17408	23.0
14	17664	82.0
15	17920	1558.0
16	18176	97.0
17	12545	145.0

페이지당 결과 수: 50 1 - 50 (전체 4362행)

작업 기록 새로고침

#### • 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 bright-meridian-439401-g6.modulabs_project1.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 bright-meridian-439401-g6.modulabs_project1.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice),0) AS user_total
  FROM bright-meridian-439401-g6.modulabs_project1.data1
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

실행 세부정보

실행 그래프

이 문으로 이름이 user\_rfm인 새 테이블이 생성되었습니다.

테이블로 이동

작업 기록

새로고침

## RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.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_rfm
1	13366	1	144	50	0.0	
2	15118	1	1440	134	0.0	
3	17752	1	192	359	0.0	
4	15316	1	100	326	2.0	
5	17896	1	288	23	2.0	
6	15313	1	25	110	2.0	
7	12791	1	96	373	2.0	
8	17775	1	72	254	2.0	
9	12814	1	48	101	2.0	
10	16953	1	10	30	2.0	
11	14705	1	100	198	2.0	
12	14124	1	1618	84	2.0	
13	17948	1	144	147	2.0	
14	18174	1	50	7	2.0	
15	17986	1	10	56	2.0	
16	12864	1	216	138	2.0	

페이지당 결과 수: 50

1 - 50 (전체 4362행)

<

>

작업 기록

새로고침

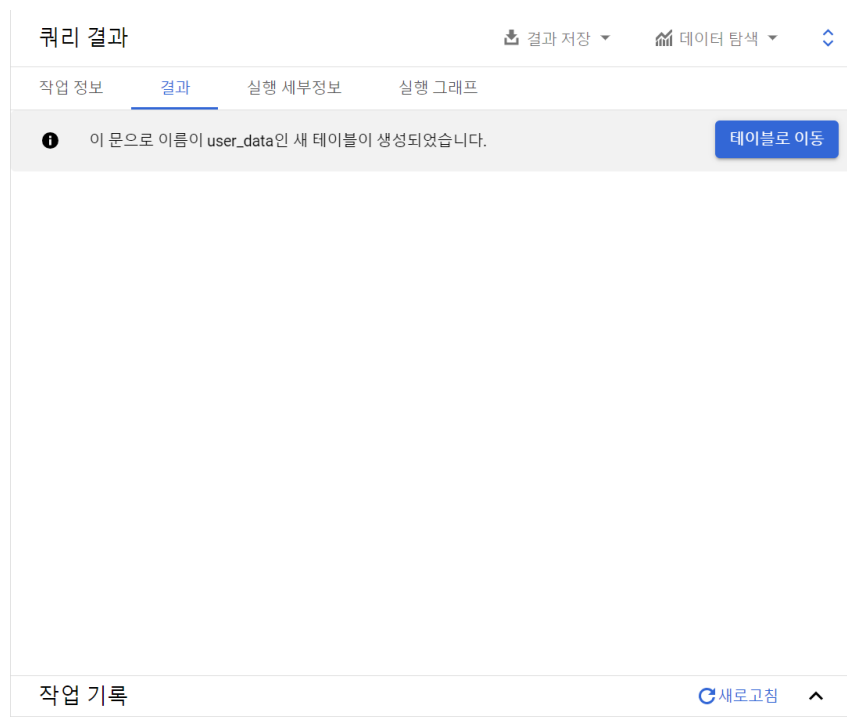
## 11-8. 추가 Feature 추출

## 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 bright-meridian-439401-g6.modulabs_project1.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
```

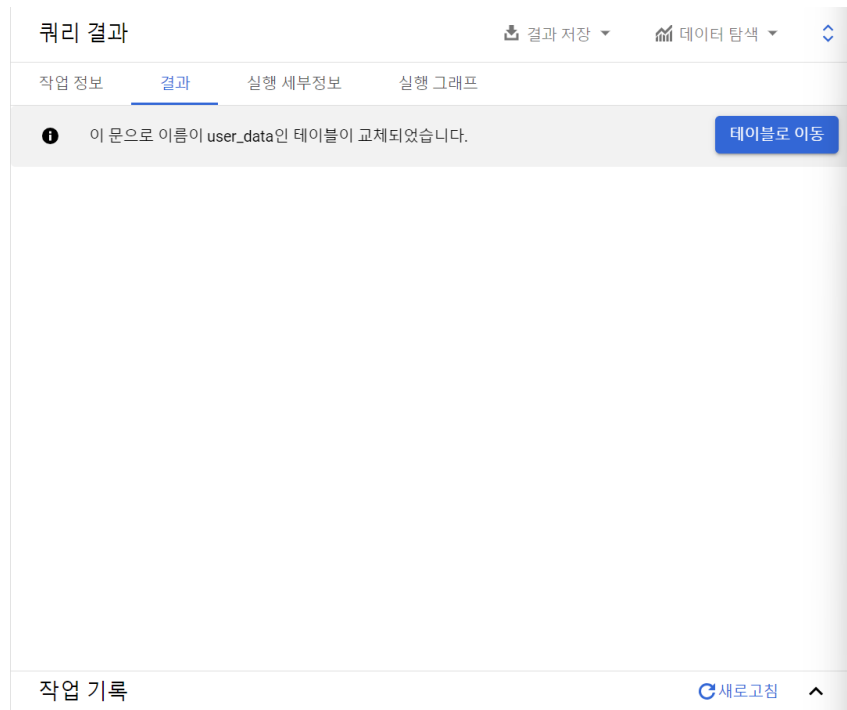
```

SELECT
    CustomerID,
    DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
FROM
    bright-meridian-439401-g6.modulabs_project1.data
WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM bright-meridian-439401-g6.modulabs_project1.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 bright-meridian-439401-g6.modulabs_project1.user_data AS

WITH TransactionInfo AS (
    SELECT
        CustomerID,
        COUNT(InvoiceNo) AS total_transactions,
        SUM(CASE WHEN stockcode LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
    FROM bright-meridian-439401-g6.modulabs_project1.data
    GROUP BY customerID
)

```

```
SELECT u.*, t.* EXCEPT(CustomerID),
       ROUND(cancel_frequency * 100.0 / total_transactions, 2)
       AS cancel_rate
FROM bright-meridian-439401-g6.modulabs_project1.user_data AS u
LEFT JOIN TransactionInfo AS t
ON U.customerID = t.customerID
```

[결과 이미지를 넣어주세요]

쿼리 결과		결과 저장	데이터 탐색	
작업 정보	결과	실행 세부정보	실행 그래프	
이 문으로 이름이 user_data인 테이블이 교체되었습니다.				테이블로 이동

- 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 **user\_data**를 출력하기

```
SELECT *
FROM bright-meridian-439401-g6.modulabs_project1.user_data
```

[결과 이미지를 넣어주세요]

쿼리 결과		결과 저장			
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt	item_cnt	recency	
1	13120	1	12		
2	15668	1	72		
3	18174	1	50		
4	14705	1	100		
5	16881	1	600		
6	16148	1	72		
7	14119	1	-2		
8	17443	1	504		
9	16953	1	10		
10	17307	1	-144		

## 회고

[회고 내용을 작성해주세요]



처음에 SQL 기초 문법을 배울때는 파이썬보다 쉽다 생각했었는데 이번 퀘스트를 하면서  
얼마나 숙지해야 할 내용들이 많고 복습이 꼭 필요하다는 것을 깨달았습니다.  
GPT나 배웠던 내용들을 보면서 여차여차 끝내긴 했는데 정말 공부를 많이 해야하고,  
쉬운 부분이 있다고 거만하지 말고, 항상 겸손한 자세가 필요하다는 것을 깨달았습니다....