



## 고객을 세그먼테이션하자 [프로젝트] - 조아영

### 11-2. 데이터 불러오기

#### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM `first-mark-479903-q6.modulabs_project.data`
LIMIT 10;
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART TUG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71058	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	844068	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)
FROM `first-mark-479903-q6.modulabs_project.data`;
```

```
6 -- 2. 데이터 수
7 SELECT COUNT(*)
8 FROM `first-mark-479903-q6.modulabs_project.data`;
```

시작하려면 쿼리를 입력하세요.

문형 처리 할당량 사용 중

#### - 쿼리 결과

f0_
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 `first-mark-479903-q6.modulabs_project.data`;<# [[YOUR QUERY]]]

```

```

9   -- 3. 결점 데이터 포인트 카운트
10  SELECT
11      COUNT(InvoiceNo) AS InvoiceNo_count,
12      COUNT(StockCode) AS StockCode_count,
13      COUNT(Description) AS Description_count,
14      COUNT(Quantity) AS Quantity_count,
15      COUNT(InvoiceDate) AS InvoiceDate_count,
16      COUNT(UnitPrice) AS UnitPrice_count,
17      COUNT(CustomerID) AS CustomerID_count,
18      COUNT(Country) AS Country_count
19
20
21 FROM `first-mark-479903-q6.modulabs_project.data`;
22
23 쿼리 완료됨
24 주문형 처리 할당량 사용 중
25
26 ← 쿼리 결과
27 작업 정보 결과 시작화 JSON 실행 세부정보 실행 그래프
28 행 InvoiceNo_count StockCode_count Description_count Quantity_count InvoiceDate_count UnitPrice_count CustomerID_count Country_count
29 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_percentage
FROM `first-mark-479903-q6.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 `first-mark-479903-q6.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 `first-mark-479903-q6.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 `first-mark-479903-q6.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 `first-mark-479903-q6.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 `first-mark-479903-q6.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 `first-mark-479903-q6.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 `first-mark-479903-q6.modulabs_project.data`;

```

제목 없는 쿼리

실행 저장

```

1 -- 컬럼 별 누락된 값의 비율 계산
2 SELECT
3     'InvoiceNo' AS column_name,
4     ROUND(
5         SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE
6             0
7         END) / COUNT(InvoiceNo), 2)
8 
```

실행 시 이 쿼리가 46.02MB를 처리합니다.

주문형 처리 할당량 사용 중

### 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보
행	column_name	missing_percent		
1	InvoiceNo	0.0		
2	StockCode	0.0		
3	Description	0.27		
4	Quantity	0.0		
5	InvoiceDate	0.0		
6	UnitPrice	0.0		
7	CustomerID	24.93		
8	Country	0.0		

### 결측치 처리 전략

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

```

SELECT Description
FROM `first-mark-479903-q6.modulabs_project.data`
WHERE StockCode = '85123A';

```

쿼리 완료됨  
주문형 처리 할당량 사용 중

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부?
행	Description			
1	WHITE HANGING HEART T-LIGH...			
2	WHITE HANGING HEART T-LIG...			
3	WHITE HANGING HEART T-LIG...			
4	WHITE HANGING HEART T-LIG...			
5	WHITE HANGING HEART T-LIG...			
6	WHITE HANGING HEART T-LIG...			
7	WHITE HANGING HEART T-LIG...			
8	WHITE HANGING HEART T-LIG...			
9	WHITE HANGING HEART T-LIG...			

## 결측치 처리

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

```
DELETE FROM `first-mark-479903-q6.modulabs_project.data`
WHERE Description IS NULL
OR CustomerID IS NULL;
```

5  
6 -- 결측치 제거  
7 DELETE FROM `first-mark-479903-q6.modulabs\_project.data`  
8 WHERE Description IS NULL  
9 OR CustomerID IS NULL;  
10

쿼리 완료됨  
주문형 처리 할당량 사용 중

← 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
1	이 문으로 data의 행 135,080개가 삭제되었습니다.		

## 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 duplicated_group_count
FROM (
    SELECT
        InvoiceNo,
        StockCode,
        Description,
        Quantity,
        InvoiceDate,
        UnitPrice,
        CustomerID,
        Country,
        COUNT(*) AS cnt
    FROM `first-mark-479903-q6.modulabs_project.data`
    GROUP BY
        InvoiceNo,
        StockCode,
        Description,
        Quantity,
        InvoiceDate,
        UnitPrice,
        CustomerID,
        Country
    HAVING COUNT(*) > 1
);
```

쿼리 결과

작업 정보	결과	人
	duplicated_group...	
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 `first-mark-479903-q6.modulabs_project.data` AS
SELECT DISTINCT *
FROM `first-mark-479903-q6.modulabs_project.data`;
```

The screenshot shows a MySQL Workbench interface. On the left, there's a sidebar with options like '설명' (Description), '라벨' (Label), '기본 키' (Primary Key), and '태그' (Tags). Below that is a section titled '스토리지 정보' (Storage Information) with various metrics such as 행수 (Rows: 401,604), 총 논리 바이트 (Total logical bytes: 34.83MB), and 현재 실제 바이트 (Current physical bytes: 2.13MB). On the right, the main area shows a query editor with the following SQL code:

```

28 -- -- 중복값 처리
29 CREATE OR REPLACE TABLE `first-mark-479903-q6.modulabs_project.data` AS
30 SELECT DISTINCT *
31 FROM `first-mark-479903-q6.modulabs_project.data`;
32
33 ✓ 쿼리 완료됨
34 주문형 처리 할당량 사용 중

```

Below the query editor is a results table with tabs for '작업 정보', '결과' (Results, which is selected), '실행 세부정보', and '실행 그래프'. A note in the results tab says: '이 문으로 이름이 data인 테이블이 교체되었습니다.'

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```

SELECT
    COUNT(DISTINCT InvoiceNo) AS unique_invoice_no_count
FROM `first-mark-479903-q6.modulabs_project.data`;

```

### 쿼리 결과

작업 정보		결과	시각화
행	unique_invoice_n...		
1	22190		

- 고유한 InvoiceNo 앞에서부터 100개를 출력하기

```

SELECT DISTINCT InvoiceNo
FROM `first-mark-479903-q6.modulabs_project.data`
ORDER BY InvoiceNo
LIMIT 100;

```

```

6 -- 100행
7 SELECT DISTINCT InvoiceNo
8 FROM `first-mark-479903-q6.modulabs_project.data`
9 ORDER BY InvoiceNo
10 LIMIT 100;

```

11 쿼리 완료됨  
주문형 처리 할당량 사용 중

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행
행	InvoiceNo				
1	536365				
2	536366				
3	536367				
4	536368				
5	536369				
6	536370				
7	536371				

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

```

SELECT *
FROM `first-mark-479903-q6.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;

```

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프			
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	Z3166	MEDIUM CERAMIC TOP STORA...	-74215	2011-03-18 10:17:00 UTC	1.04	12346	United Kingdom
2	CS45329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
3	CS45329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	CS45330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	CS47388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
6	CS47388	22413	METAL SIGN TAKE IT OR LEAVE...	-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 ELSE 0 END)
  / COUNT(*) * 100,
  1
) AS canceled_percentage
FROM `first-mark-479903-q6.modulabs_project.data`;

```

← 쿼리 결과

작업 정보	결과	시각화	JSON
행	cancelled_percent...		
1	2.2		

## StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
-- StockCode 개수
SELECT
    COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `first-mark-479903-q6.modulabs_project.data`;
```

The screenshot shows a query editor interface with the following details:

- Query Text:**

```
1 -- StockCode 개수
2 SELECT
3     COUNT(DISTINCT StockCode) AS unique_stockcode_count
4 FROM `first-mark-479903-q6.modulabs_project.data`;
```
- Status Bar:** 쿼리 완료됨 (Query completed), 주문형 처리 할당량 사용 중 (Order-type processing usage in progress).
- Result Tab:** 결과 (Results) is selected.
- Result Data:**

unique_stockcode_count
3684

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 **StockCode** 별 등장 빈도를 출력하기

- 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `first-mark-479903-q6.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

The screenshot shows a query editor interface with the following details:

- Query Text:**

```
6 -- 상위 10개의 제품들을 출력하기
7 SELECT StockCode, COUNT(*) AS sell_cnt
8 FROM `first-mark-479903-q6.modulabs_project.data`
9 GROUP BY StockCode
10 ORDER BY sell_cnt DESC
11 LIMIT 10;
12
```
- Status Bar:** 실행 시 이 스크립트가 5.42MB를 처리합니다. (Execution of this script will process 5.42MB), 주문형 처리 할당량 사용 중 (Order-type processing usage in progress).
- Result Tab:** 결과 (Results) is selected.
- Result Data:**

StockCode	sell_cnt
85123A	2065
22423	1894
85099B	1659
47566	1409
84879	1405

- **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 `first-mark-479903-q6.modulabs_project.data`
)
WHERE number_count <= 1;

```

15  
16 SELECT DISTINCT StockCode, number\_count  
17 FROM (  
18 SELECT  
19 StockCode,  
20 LENGTH(StockCode) - LENGTH(REGEXP\_REPLACE(StockCode, r'[0-9]', '')) AS  
21 FROM `first-mark-479903-q6.modulabs\_project.data`  
22 )  
23 WHERE number\_count <= 1;  
24

쿼리 완료됨  
주문형 처리 할당량 사용 중

[← 쿼리 결과](#)

작업 정보	결과	시각화	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			

- 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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```

SELECT DISTINCT StockCode, number_count
FROM (
  SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `first-mark-479903-q6.modulabs_project.data`  

)
WHERE number_count <= 1;

```

40 FROM (  
41 SELECT  
42 StockCode,  
43 LENGTH(StockCode) - LENGTH(REGEXP\_REPLACE(StockCode, r'[0-9]', '')) AS  
44 FROM `first-mark-479903-q6.modulabs\_project.data`  
45 )  
46 WHERE number\_count <= 1;  
47  
48 -- 해당 코드 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트  
49 실행 시 이 쿼리가 2.71MB를 처리합니다.  
50 주문형 처리 할당량 사용 중

[← 쿼리 결과](#)

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	percentage_0_1...				
1	0.48				

- 제품과 관련되지 않은 거래 기록을 제거하기

```

DELETE FROM `first-mark-479903-q6.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 `first-mark-479903-q6.modulabs_project.data`
    )
    WHERE number_count <= 1
);

```

The screenshot shows a code editor with a SQL query and its execution results. The query is identical to the one above, with line numbers 61 to 72. A checkmark indicates the query was executed successfully. Below the code, a message says "주문형 처리 할당량 사용 중". The results section shows a single row: "이 문으로 data의 행 1,915개가 삭제되었습니다." (1,915 rows deleted by this query).

```

61 -- 제품과 관련되지 않은 거래 기록을 제거하기
62 DELETE FROM `first-mark-479903-q6.modulabs_project.data`
63 WHERE StockCode IN (
64     SELECT DISTINCT StockCode
65     FROM (
66         SELECT
67             StockCode,
68             LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
69         FROM `first-mark-479903-q6.modulabs_project.data`
70     )
71     WHERE number_count <= 1
72 )

```

✓ 쿼리 완료됨

주문형 처리 할당량 사용 중

← 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
	<span>❶</span> 이 문으로 data의 행 1,915개가 삭제되었습니다.		

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```

SELECT Description, COUNT(*) AS description_cnt
FROM `first-mark-479903-q6.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;

```

```

1 -- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기
2 SELECT Description, COUNT(*) AS description_cnt
3 FROM `first-mark-479903-q6.modulabs_project.data`
4 GROUP BY Description
5 ORDER BY description_cnt DESC
6 LIMIT 30;
7

```

쿼리 완료됨  
주문형 처리 할당량 사용 중

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
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			

- 서비스 관련 정보를 포함하는 행들을 제거하기

```

DELETE
FROM `first-mark-479903-q6.modulabs_project.data`
WHERE
    Description LIKE '%POST%'
    OR Description LIKE '%BANK%'
    OR Description LIKE '%CHARGE%'
    OR Description LIKE '%CARRIAGE%'
    OR Description LIKE '%ADJUST%'
    OR Description LIKE '%SAMPLE%'
    OR Description LIKE '%CHECK%';

```

```

7 -- 서비스 관련 정보를 포함하는 행들을 제거하기
8 DELETE
9 FROM `first-mark-479903-q6.modulabs_project.data`
10 WHERE
11     Description LIKE '%POST%'
12     OR Description LIKE '%BANK%'
13     OR Description LIKE '%CHARGE%'
14     OR Description LIKE '%CARRIAGE%'
15     OR Description LIKE '%ADJUST%'
16     OR Description LIKE '%SAMPLE%'
17     OR Description LIKE '%CHECK%';

```

쿼리 완료됨  
주문형 처리 할당량 사용 중

← 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
1	이 문으로 data의 행 1,992개가 삭제되었습니다.		

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```

CREATE OR REPLACE TABLE `first-mark-479903-q6.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER>Description) AS Description
FROM `first-mark-479903-q6.modulabs_project.data`;

```

The screenshot shows the BigQuery interface with the following code in the query editor:

```

18 -- Description 칸에 소문자(a-z)가 포함된 행만 찾기
19 SELECT DISTINCT Description
20 FROM `first-mark-479903-q6.modulabs_project.data`
21 WHERE REGEXP_CONTAINS>Description, r'[a-z]');
22
23 -- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화
24 CREATE OR REPLACE TABLE `first-mark-479903-q6.modulabs_project.data` AS
25 SELECT
26   * EXCEPT (Description),
27   UPPER>Description) AS Description
28 FROM `first-mark-479903-q6.modulabs_project.data`;
29

```

Below the code, a note says "실행 시 이 스크립트가 90.87MB를 처리합니다." (Execution will process 90.87MB). The status bar indicates "주문형 처리 할당량 사용 중" (Using quota for on-demand processing).

At the bottom, there's a "쿼리 결과" (Query Results) section with tabs for "작업 정보", "결과" (selected), "시작화", "JSON", "실행 세부정보", and "실행 그래프". A note says "이 문으로 이름이 data인 테이블이 교체되었습니다." (This statement overwrites a table named 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 `first-mark-479903-q6.modulabs_project.data`;

```

The screenshot shows the BigQuery interface with the following code in the query editor:

```

1 -- UnitPrice 의 최솟값, 최댓값, 평균을 구하기
2 SELECT
3   MIN(UnitPrice) AS min_price,
4   MAX(UnitPrice) AS max_price,
5   AVG(UnitPrice) AS avg_price
6 FROM `first-mark-479903-q6.modulabs_project.data`;
7

```

Below the code, a note says "쿼리 완료됨" (Query completed). The status bar indicates "주문형 처리 할당량 사용 중" (Using quota for on-demand processing).

At the bottom, there's a "쿼리 결과" (Query Results) section with tabs for "작업 정보", "결과" (selected), "시작화", "JSON", "실행 세부정보", and "실행 그래프". The results table shows:

	min_price	max_price	avg_price
1	0.0	649.5	2.910880921907...

- 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 `first-mark-479903-q6.modulabs_project.data`
WHERE UnitPrice = 0;

```

```

/
8 -- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기
9 SELECT
10 COUNT(*) AS cnt_quantity,
11 MIN(Quantity) AS min_quantity,
12 MAX(Quantity) AS max_quantity,
13 AVG(Quantity) AS avg_quantity
14 FROM `first-mark-479903-q6.modulabs_project.data`
15 WHERE UnitPrice = 0;
16
17 쿼리 완료됨
18 주문형 처리 할당량 사용 중
19
20 ← 쿼리 결과
21
22 작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프
23
24 행 cnt_quantity min_quantity max_quantity avg_quantity
25 1 33 1 12540 420.5151515151515

```

- `UnitPrice = 0` 를 제거하고 일관된 데이터셋을 유지하기

```

CREATE OR REPLACE TABLE `first-mark-479903-q6.modulabs_project.data` AS
SELECT *
FROM `first-mark-479903-q6.modulabs_project.data`
WHERE UnitPrice != 0;

```

```

/
7
8 -- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기
9 SELECT
10 COUNT(*) AS cnt_quantity,
11 MIN(Quantity) AS min_quantity,
12 MAX(Quantity) AS max_quantity,
13 AVG(Quantity) AS avg_quantity
14 FROM `first-mark-479903-q6.modulabs_project.data`
15 WHERE UnitPrice = 0;
16 -- UnitPrice = 0를 제거하고 일관된 데이터셋을 유지하기
17 CREATE OR REPLACE TABLE `first-mark-479903-q6.modulabs_project.data` AS
18 SELECT *
19
20 쿼리 완료됨
21 주문형 처리 할당량 사용 중
22
23 ← 쿼리 결과
24
25 작업 정보 결과 실행 세부정보 실행 그래프
26
27 ❶ 이 문으로 이름이 data인 테이블이 교체되었습니다.

```

## 11-7. RFM 스코어

### Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```

SELECT
    DATE(InvoiceDate) AS InvoiceDay,
    *
FROM `first-mark-479903-q6.modulabs_project.data`;

```

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
#	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-01-18	541431	23166	74215	2011-01-18 10:00:00 UTC
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC
3	2010-12-07	537626	22775	12	2010-12-07 14:57:00 UTC
4	2010-12-07	537626	20782	6	2010-12-07 14:57:00 UTC
5	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC
6	2010-12-07	537626	22728	4	2010-12-07 14:57:00 UTC
7	2010-12-07	537626	21171	12	2010-12-07 14:57:00 UTC
8	2010-12-07	537626	21064	6	2010-12-07 14:57:00 UTC
9	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC
10	2011-01-18	493749A	99999	17	2011-01-18 10:47:00 UTC
					1.04
					1.25
					5.49
					0.65
					3.75
					1.45
					5.95
					2.95
					1.95
					19347
					Iceland

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```

SELECT
    MAX(InvoiceDate) OVER () AS most_recent_date,
    DATE(InvoiceDate) AS InvoiceDay,
    *
FROM `first-mark-479903-q6.modulabs_project.data`;

```

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
#	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity
1	2011-12-09 12:50:00 UTC	2011-01-18	541431	23166	74215
2	2011-12-09 12:50:00 UTC	2011-01-18	C541433	23166	-74215
3	2011-12-09 12:50:00 UTC	2010-12-07	537626	22775	12
4	2011-12-09 12:50:00 UTC	2010-12-07	537626	20782	6
5	2011-12-09 12:50:00 UTC	2010-12-07	537626	22492	36
6	2011-12-09 12:50:00 UTC	2010-12-07	537626	22728	4
7	2011-12-09 12:50:00 UTC	2010-12-07	537626	21171	12

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```

SELECT
    CustomerID,
    # [[YOUR QUERY]] AS InvoiceDay
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];

```

```

14 -- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하
15 SELECT
16   CustomerID,
17   MAX(DATE(InvoiceDate)) AS InvoiceDay
18 FROM `first-mark-479903-q6.modulabs_project.data`
```

✓ 쿼리 완료됨

주문형 처리 할당량 사용 중

◀ 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행
행	CustomerID	InvoiceDay			
1	12346	2011-01-18			
2	12347	2011-12-07			
3	12348	2011-09-25			
4	12349	2011-11-01			

- 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
);
```

◀ 쿼리 결과

작업 정보	결과	시각화	JSON	슬
행	CustomerID	recency		
1	12609	78		
2	12716	3		
3	12744	56		
4	12906	11		
5	12956	306		
6	13130	94		
7	13261	268		

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

```

CREATE OR REPLACE TABLE `first-mark-479903-q6.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 `first-mark-479903-q6.modulabs_project.data`
```

```
GROUP BY CustomerID  
);
```

## ← 쿼리 결과

작업 정보    **결과**    실행 세부정보    실행 그래프

❶ 이 문으로 이름이 user\_r인 새 테이블이 생성되었습니다.

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT  
CustomerID,  
COUNT(DISTINCT InvoiceNo) AS purchase_cnt  
FROM `first-mark-479903-q6.modulabs_project.data`  
GROUP BY CustomerID;
```

```
1 -- Frequency  
2 -- 고객마다 고유한 InvoiceNo의 수를 세어보기  
3 SELECT  
4   CustomerID,  
5   COUNT(DISTINCT InvoiceNo) AS purchase_cnt  
6   FROM `first-mark-479903-q6.modulabs_project.data`  
7   GROUP BY CustomerID;
```

✅ 쿼리 완료됨

주문형 처리 할당량 사용 중

## 쿼리 결과

작업 정보    **결과**    시각화    JSON    실행 세부정보

행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT  
CustomerID,  
SUM(Quantity) AS item_cnt  
FROM `first-mark-479903-q6.modulabs_project.data`  
GROUP BY CustomerID;
```

행	CustomerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	439
7	12353	20
8	12354	530
9	12355	240
10	12356	1561
11	12357	2708
12	12358	206
13	12359	1599

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

```

CREATE OR REPLACE TABLE `first-mark-479903-q6.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `first-mark-479903-q6.modulabs_project.data`
GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
FROM `first-mark-479903-q6.modulabs_project.data`
GROUP BY CustomerID
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
    pc.CustomerID,
    pc.purchase_cnt,
    ic.item_cnt,
    ur.recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
    ON pc.CustomerID = ic.CustomerID
JOIN `first-mark-479903-q6.modulabs_project.user_r` AS ur
    ON pc.CustomerID = ur.CustomerID;

```

```

9  -- (1) 전체 거래 건수 계산
0 WITH purchase_cnt AS (
1   SELECT
2     CustomerID,
3     COUNT(DISTINCT InvoiceNo) AS purchase_cnt
4   FROM `first-mark-479903-q6.modulabs_project.data`
5   GROUP BY CustomerID
6 ),
7
8  -- (2) 구매한 아이템 총 수량 계산
9 item_cnt AS (
0   SELECT
1     CustomerID,
2     SUM(Quantity) AS item_cnt

```

쿼리 완료됨

문형 처리 할당량 사용 중

#### - 쿼리 결과

|업 정보      결과      실행 세부정보      실행 그래프

이 문으로 이름이 user\_rf인 새 테이블이 생성되었습니다.

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `first-mark-479903-q6.modulabs_project.data`
GROUP BY CustomerID;

```

작업 정보	결과	시각화	JSON	설명
행	CustomerID	user_total		
1	12346	0.0		
2	12347	4310.0		
3	12348	1437.2		
4	12349	1457.5		
5	12350	294.4		
6	12352	1230.6		
7	12353	89.0		
8	12354	1079.4		
9	12355	459.4		
10	12356	2470.0		
11	12357	6207.7		
12	12358	914.0		
13	12359	6182.8		

- 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 `first-mark-479903-q6.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 `first-mark-479903-q6.modulabs_project.user_rf` rf
LEFT JOIN (
-- 고객 별 총 지출액
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `first-mark-479903-q6.modulabs_project.data`
GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;

```

▶ 쿼리 완료됨  
주문형 처리 할당량 사용 중

◀ 쿼리 결과

작업 정보      결과      실행 세부정보      실행 그래프

ℹ 이 문으로 이름이 user\_rfm인 새 테이블이 생성되었습니다.

## RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```

SELECT *
FROM `first-mark-479903-q6.modulabs_project.user_rfm`;

```

◀ 쿼리 결과

작업 정보      결과      시각화      JSON      실행 세부정보      실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	480	0	784.0	784.0
2	15520	1	314	1	343.5	343.5
3	14569	1	79	1	227.4	227.4
4	13436	1	72	1	181.9	181.9
5	13298	1	96	1	360.0	360.0
6	15471	1	255	2	447.8	447.8
7	15195	1	1404	2	3861.0	3861.0
8	14204	1	72	2	150.6	150.6
9	15318	1	606	3	298.6	298.6

## 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 `first-mark-479903-q6.modulabs_project.user_data` AS
WITH unique_products AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT StockCode) AS unique_products
    FROM `first-mark-479903-q6.modulabs_project.data`
    GROUP BY CustomerID
)
SELECT
    ur.*,
    up.* EXCEPT (CustomerID)
FROM `first-mark-479903-q6.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

❖ Gemini로 테이블 및 열 설명을 생성하세요. 프로필 스캔을 통해 설명을

≡ 필터 속성 이름 또는 값 입력

<input type="checkbox"/>	필드 이름	유형	모드	설명
<input type="checkbox"/>	CustomerID	INTEGER	NULLABLE	-
<input type="checkbox"/>	purchase_cnt	INTEGER	NULLABLE	-
<input type="checkbox"/>	item_cnt	INTEGER	NULLABLE	-
<input type="checkbox"/>	recency	INTEGER	NULLABLE	-
<input type="checkbox"/>	user_total	FLOAT	NULLABLE	-
<input type="checkbox"/>	user_average	FLOAT	NULLABLE	-
<input type="checkbox"/>	unique_products	INTEGER	NULLABLE	-

## 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 `first-mark-479903-q6.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(
```

```

DATE(InvoiceDate),
LAG(DATE(InvoiceDate)) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate),
DAY
) AS interval_
FROM
`first-mark-479903-q6.modulabs_project.data`
WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

SELECT
u.*,
pi.* EXCEPT (CustomerID)
FROM `first-mark-479903-q6.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

```

쿼리 완료됨  
주문형 처리 할당량 사용 중

← 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
	<span>ⓘ</span> 이 문으로 이름이 user_data인 데이터가 교체되었습니다.		

### 3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기
  - 취소 빈도(cancel\_frequency) : 고객 별로 취소한 거래의 총 횟수
  - 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 `first-mark-479903-q6.modulabs_project.user_data` AS

WITH TransactionInfo AS (
SELECT
CustomerID,
COUNT(*) AS total_transactions,
SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
FROM `first-mark-479903-q6.modulabs_project.data`
GROUP BY CustomerID
)

SELECT
u.*,
t.* EXCEPT (CustomerID),

```

```

ROUND(
  IFNULL(t.cancel_frequency, 0) / NULLIF(t.total_transactions, 0) * 100,
  2
) AS cancel_rate
FROM `first-mark-479903-q6.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;

```

The screenshot shows the BigQuery schema editor interface for the `user_data` table. The top navigation bar includes tabs for `user_data`, `쿼리` (Query), `다음에서 열기` (Open Next), `공유` (Share), and `복사` (Copy). Below the navigation is a toolbar with `스키마` (Schema), `세부정보` (Details), `미리보기` (Preview), `테이블 탐색기` (Table Browser), `프리뷰` (Preview) (which is selected), `통계` (Statistics), and `계보` (Lineage).

**필터** section: `속성 이름 또는 값 입력`

<input type="checkbox"/> 필드 이름	유형	모드	설명
<input type="checkbox"/> CustomerID	INTEGER	NULLABLE	-
<input type="checkbox"/> purchase_cnt	INTEGER	NULLABLE	-
<input type="checkbox"/> item_cnt	INTEGER	NULLABLE	-
<input type="checkbox"/> recency	INTEGER	NULLABLE	-
<input type="checkbox"/> user_total	FLOAT	NULLABLE	-
<input type="checkbox"/> user_average	FLOAT	NULLABLE	-
<input type="checkbox"/> unique_products	INTEGER	NULLABLE	-

Buttons at the bottom: `스키마 설정` (Schema Settings) and `행 액세스 정책 보기` (View Row Access Policy).

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

```

SELECT *
FROM `first-mark-479903-q6.modulabs_project.user_data`;

```

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
명	CustomerID	purchase_cnt	item_cnt	recency	user_total
1	13607	1	288	40	675.5
2	14675	1	336	16	596.4
3	16776	1	254	60	371.5
4	12371	1	582	59	1528.0
5	15472	1	251	113	371.6
6	17914	1	451	3	326.0
7	17832	1	202	49	153.0
8	16445	1	110	33	226.8
9	17070	1	131	114	304.2
10	14954	1	109	12	281.3
11	15004	1	622	147	1220.4
12	18082	1	600	25	669.2

## 회고

[회고 내용을 작성해주세요]

Keep : BigQuery 환경에서 차근차근 SQL 실습을 진행하였다.

Problem : 윈도우 함수, DATE 처리, 조건 조합에서 개념과 문법이 헷갈렸다.

Try : 틀린 쿼리는 바로 고치기보다 어디서부터 틀렸는지 단계별로 검증하는 습관 만들기