



## 고객을 세그먼테이션하자 [프로젝트] - 김수경

### 11-2. 데이터 불러오기

#### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
select *  
from `project-dac3b643-036f-483e-a72.modulabs_project.data`  
limit 10;
```

| 쿼리 결과 |    |     |      |         |        |    |           |           |                              |
|-------|----|-----|------|---------|--------|----|-----------|-----------|------------------------------|
| 작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프 | 행  | InvoiceNo | StockCode | Description                  |
|       |    |     |      |         |        | 1  | 536365    | 85123A    | WHITE HANING HEART TLIQ...   |
|       |    |     |      |         |        | 2  | 536365    | 71053     | WHITE METAL LANTERN          |
|       |    |     |      |         |        | 3  | 536365    | 84406B    | CREAM CUPID HEARTS COAT H... |
|       |    |     |      |         |        | 4  | 536365    | 840295    | KNITTED UNION FLAG HOT WA... |
|       |    |     |      |         |        | 5  | 536365    | 84029E    | RED WOOLY HOTTIE WHITE H...  |
|       |    |     |      |         |        | 6  | 536365    | 22752     | SET 7 BABUSHKA NESTING BO... |
|       |    |     |      |         |        | 7  | 536365    | 21730     | GLASS STAR FROSTED TLIGHT... |
|       |    |     |      |         |        | 8  | 536366    | 22633     | HAND WARMER UNION JACK       |
|       |    |     |      |         |        | 9  | 536366    | 22632     | HAND WARMER RED POLKA DOT    |
|       |    |     |      |         |        | 10 | 536367    | 84879     | ASORTED COLOUR BIRD ORN...   |

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
select count(*)  
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

| 쿼리 결과 |          |
|-------|----------|
| 작업 정보 | 결과       |
|       | 1 541909 |

- 541,909행

#### 데이터 수 세기

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

```

select count(invoiceno) as Invoicenos,
       count(stockcode) as Stockcodes,
       count(description) as Descriptions,
       count(quantity) as Quantities,
       count(invoicedate) as Invoicedates,
       count(unitprice) as Unitprices,
       count(customerid) as Customerids,
       count(country) as Countries
  from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

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

- Query Text:**

```

1 select count(invoiceno) as Invoicenos,
2       count(stockcode) as Stockcodes,
3       count(description) as Descriptions,
4       count(quantity) as Quantities,
5       count(invoicedate) as Invoicedates,
6       count(unitprice) as Unitprices,
7       count(customerid) as Customerids,
8       count(country) as Countries
9  from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```
- Execution Status:** 실행 (Running)
- Result Preview:**

|   | Invoicenos | Stockcodes | Descriptions | Quantities | Invoicedates | Unitprices | Customerids | Countries |
|---|------------|------------|--------------|------------|--------------|------------|-------------|-----------|
| 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(COUNTIF(InvoiceNo IS NULL) / COUNT(*) * 100, 2) AS missing_percentage
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'StockCode',
       ROUND(COUNTIF(StockCode IS NULL) / COUNT(*) * 100, 2)
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'Description',
       ROUND(COUNTIF(Description IS NULL) / COUNT(*) * 100, 2)
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'Quantity',
       ROUND(COUNTIF(Quantity IS NULL) / COUNT(*) * 100, 2)
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'InvoiceDate',
       ROUND(COUNTIF(InvoiceDate IS NULL) / COUNT(*) * 100, 2)
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'UnitPrice',
       ROUND(COUNTIF(UnitPrice IS NULL) / COUNT(*) * 100, 2)
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'CustomerID',
       ROUND(COUNTIF(CustomerID IS NULL) / COUNT(*) * 100, 2)
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
UNION ALL
SELECT 'Country',
```

```
ROUND(COUNTIF(Country IS NULL) / COUNT(*) * 100, 2)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;
```

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

- Query Text:**

```
1 SELECT 'InvoiceNo' AS column_name,
2      |   ROUND(COUNTIF(InvoiceNo IS NULL) / COUNT(*) * 100, 2) AS missing_percentage
3 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
4 UNION ALL
5 SELECT 'StockCode',
6      |   ROUND(COUNTIF(StockCode IS NULL) / COUNT(*) * 100, 2)
7 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
8 UNION ALL
9 SELECT 'Description',
10    |   ROUND(COUNTIF(Description IS NULL) / COUNT(*) * 100, 2)
11 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
12 UNION ALL
13 SELECT 'Quantity',
14      |   ROUND(COUNTIF(Quantity IS NULL) / COUNT(*) * 100, 2)
```
- Execution Status:** The status bar at the bottom indicates "실행 시 이 쿼리가 46.02MB를 처리합니다." (Executing this query processes 46.02MB).
- Results:** The results table shows the missing percentage for each column:

| column_name | missing_percentage |
|-------------|--------------------|
| InvoiceNo   | 0.0                |
| StockCode   | 0.0                |
| Description | 0.27               |
| Quantity    | 0.0                |
| InvoiceDate | 0.0                |
| UnitPrice   | 0.0                |
| CustomerID  | 24.93              |
| Country     | 0.0                |

## 결측치 처리 전략

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

```
SELECT Description
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE StockCode = '85123A'
GROUP BY Description
```

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

- Query Text:**

```
1 SELECT Description
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
3 WHERE StockCode = '85123A'
4 GROUP BY Description
```
- Execution Status:** A progress bar indicates "주문형 처리 할당량 사용 중" (Using assigned quantity for ordered processing).

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

- Results:** The results table shows the descriptions for StockCode '85123A':

| Description                  |
|------------------------------|
| WHITE HANGING HEART T-LIG... |
| ?                            |
| wrongly marked carton 22804  |
| CREAM HANGING HEART T-LIG... |

## 결측치 처리 (⭐ delete 후 이전 원본 데이터를 재활용하기 )

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

```
DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`  
WHERE Customerid is null  
or Description is null
```

The screenshot shows a MySQL query editor interface. At the top, there is a toolbar with icons for '실행' (Execute), '쿼리 저장' (Save Query), '다운로드' (Download), and '공유' (Share). The main area contains the SQL code:

```
1 DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`  
2 WHERE Customerid is null  
3 or Description is null
```

Below the code, a progress bar indicates '주문형 처리 할당량 사용 중' (Using allocation for ordered processing). The results section shows the message: '이 문으로 data의 행 135,080개가 삭제되었습니다.' (135,080 rows of data were deleted by this statement).

## 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 cnt  
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`  
group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country  
having count(*) >= 2
```

The screenshot shows a MySQL query editor interface. At the top, there is a toolbar with icons for '실행' (Execute), '쿼리 저장' (Save Query), '다운로드' (Download), '공유' (Share), '일정' (Schedule), '다음에서 열기' (Open next), and '더보기' (More). The main area contains the SQL code from the previous step.

Below the code, a progress bar indicates '실행 시 이 쿼리가 35.29MB를 처리합니다.' (This query processes 35.29MB when executed) and '주문형 처리 할당량 사용 중' (Using allocation for ordered processing).

The results section shows a table with columns: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, and cnt. The table lists various items with their counts. For example, item 571034 has a count of 2, and item 577228 has a count of 12.

| InvoiceNo | StockCode | Description                    | Quantity | InvoiceDate             | UnitPrice | CustomerID | Country   | cnt |
|-----------|-----------|--------------------------------|----------|-------------------------|-----------|------------|-----------|-----|
| 1         | 571034    | SET OF 4 KNOCK KNUCK TINS P... | 6        | 2011-10-10 12:47:00 UTC | 4.15      | 12359      | Cyprus    | 2   |
| 2         | 571034    | VINTAGE DOILY DELUXE SEWIN...  | 3        | 2011-10-10 12:47:00 UTC | 5.95      | 12359      | Cyprus    | 2   |
| 3         | 571034    | SET OF 3 REGENCY CAKE TINS     | 4        | 2011-10-10 12:47:00 UTC | 4.95      | 12359      | Cyprus    | 2   |
| 4         | 538826    | FELTCRAFT PRINCESS CHARLO...   | 1        | 2010-12-14 12:58:00 UTC | 3.75      | 12370      | Cyprus    | 2   |
| 5         | 577228    | SET OF 5 MINI GROCERY MAG...   | 1        | 2011-11-18 12:07:00 UTC | 2.08      | 12391      | Cyprus    | 2   |
| 6         | 577228    | MOUSE TOY WITH PINK TSHIRT     | 1        | 2011-11-18 12:07:00 UTC | 3.75      | 12391      | Cyprus    | 2   |
| 7         | 577228    | SET OF 4 HEART SHAPED BALL...  | 1        | 2011-11-18 12:07:00 UTC | 1.25      | 12391      | Cyprus    | 2   |
| 8         | 577228    | CHRISTMAS CRAFT LITTLE FRI...  | 1        | 2011-11-18 12:07:00 UTC | 2.1       | 12391      | Cyprus    | 2   |
| 9         | 577228    | SET OF 10 LANTERNS FAIRY LI... | 1        | 2011-11-18 12:07:00 UTC | 4.15      | 12391      | Cyprus    | 3   |
| 10        | 577228    | HAPPY EASTER HANGING DEC...    | 1        | 2011-11-18 12:07:00 UTC | 3.75      | 12391      | Cyprus    | 2   |
| 11        | 589419    | DOORMAT UNION FLAG             | 10       | 2016-12-17 14:10:00 UTC | 6.75      | 12431      | Australia | 2   |
| 12        | 588174    | ROUND SNACK BOXES SET OF 4...  | 12       | 2016-12-10 09:35:00 UTC | 2.95      | 12471      | Germany   | 2   |

페이지당 결과 수: 50 | 1 ~ 50 (전체 48378개) | < > >>

## 중복값 처리

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

```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` as  
select distinct *  
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

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

- Query ID: project\_mainquest\_03
- Buttons: 실행 (Run), 쿼리 저장 (Save Query), 다운로드 (Download), 공유 (Share)
- Query Text:

```
1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` as  
2 select distinct *  
3 from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```
- Status: 쿼리 완료됨 (Query completed)
- Message: 주문형 처리 할당량 사용 중 (Using allocation for ordered processing)
- Result Tab: 결과 (Results) is selected.
- Message under Results: ⓘ 이 문으로 이름이 data인 테이블이 교체되었습니다. (A table named data was replaced by this query.)

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
select count(invoiceno)  
from `project-dac3b643-036f-483e-a72.modulabs_project.data`  
group by invoiceno
```

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

- Query ID: project\_mainquest\_03
- Buttons: 실행 (Run), 쿼리 저장 (Save Query), 다운로드 (Download), 공유 (Share), 일정 (Schedule), 다음에서 열기 (Open Next), 더보기 (More)
- Query Text:

```
1 select count(invoiceno)  
2 from `project-dac3b643-036f-483e-a72.modulabs_project.data`  
3 group by invoiceno
```
- Status: 쿼리 완료됨 (Query completed)
- Result Tab: 결과 (Results) is selected.
- Data Table:

| 명  | invoiceno |
|----|-----------|
| 1  | 541431    |
| 2  | C541433   |
| 3  | 537626    |
| 4  | 542237    |
| 5  | 549222    |
| 6  | 556201    |
| 7  | 562032    |
| 8  | 573511    |
| 9  | 581180    |
| 10 | 539318    |
| 11 | 541998    |
| 12 | 548955    |
| 13 | 564172    |
| 14 | 577609    |
| 15 | 543037    |
- Page Footer: 페이지당 결과 수: 50 ▾ 1 ~ 50 (전체 221900개) | < > >>

- 고유한 InvoiceNo를 앞에서부터 100개를 출력하기

```
select invoiceno  
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

```
group by invoceno  
order by invoceno  
limit 100
```

project\_mainquest\_03 [실행] [쿼리 저장] [다운로드] [공유] [일정] [다음에서 열기] [더보기]

주문형 처리 활동량 사용 중

쿼리 결과

| 작업 정보 | 결과       | 시작화 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|----------|-----|------|---------|--------|
| 명     | invoceno |     |      |         |        |
| 1     | S06365   |     |      |         |        |
| 2     | S06366   |     |      |         |        |
| 3     | S06367   |     |      |         |        |
| 4     | S06368   |     |      |         |        |
| 5     | S06369   |     |      |         |        |
| 6     | S06370   |     |      |         |        |
| 7     | S06371   |     |      |         |        |
| 8     | S06372   |     |      |         |        |
| 9     | S06373   |     |      |         |        |
| 10    | S06374   |     |      |         |        |
| 11    | S06375   |     |      |         |        |
| 12    | S06376   |     |      |         |        |
| 13    | S06377   |     |      |         |        |
| 14    | S06378   |     |      |         |        |
| 15    | S06380   |     |      |         |        |

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

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

```
select invoceno  
from `project-dac3b643-036f-483e-a72.modulabs_project.data`  
where invoceno like 'C%'  
group by invoceno  
order by invoceno  
limit 100
```

project\_mainquest\_03 [실행] [쿼리 저장] [다운로드] [공유] [일정] [다음에서 열기] [더보기]

주문형 처리 활동량 사용 중

쿼리 결과

| 작업 정보 | 결과       | 시작화 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|----------|-----|------|---------|--------|
| 명     | invoceno |     |      |         |        |
| 1     | CS06379  |     |      |         |        |
| 2     | CS06383  |     |      |         |        |
| 3     | CS06391  |     |      |         |        |
| 4     | CS06306  |     |      |         |        |
| 5     | CS06543  |     |      |         |        |
| 6     | CS06548  |     |      |         |        |
| 7     | CS06606  |     |      |         |        |
| 8     | CS06622  |     |      |         |        |
| 9     | CS06625  |     |      |         |        |
| 10    | CS06642  |     |      |         |        |
| 11    | CS06734  |     |      |         |        |
| 12    | CS06737  |     |      |         |        |
| 13    | CS06757  |     |      |         |        |
| 14    | CS06768  |     |      |         |        |
| 15    | CS06760  |     |      |         |        |

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

- 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 InvoiceNo  
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

```

@ project_mainquest_03
▶ 실행   ⌂ 쿼리 저장   ⌂ 다운로드   ⌂ 공유   ⓘ 일정   다음에서 열기 ▾
1 SELECT ROUND(SUM(CASE WHEN InvoiceNo like 'C%' THEN 1 ELSE 0 END)/count(*) * 100, 1) as InvoiceNo
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`

```

주문형 처리 할당량 사용 중

쿼리 결과

| 작업 정보 | 결과        | 시각화 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|-----------|-----|------|---------|--------|
|       | InvoiceNo |     |      |         |        |
| 1     | 2.2       |     |      |         |        |

## StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```

SELECT count(stockcode)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
GROUP BY stockcode

```

```

@ project_mainquest_03
▶ 실행   ⌂ 쿼리 저장   ⌂ 다운로드   ⌂ 공유   ⓘ 일정   다음에서 열기 ▾   ⌂ 더보기 ▾
1 SELECT count(stockcode)
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
3 GROUP BY stockcode

```

실행 시 이 쿼리가 2.71MB를 차리합니다.

쿼리 결과

| 작업 정보 | 결과  | 시각화 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|-----|-----|------|---------|--------|
|       | id  |     |      |         |        |
| 1     | 208 |     |      |         |        |
| 2     | 399 |     |      |         |        |
| 3     | 18  |     |      |         |        |
| 4     | 226 |     |      |         |        |
| 5     | 539 |     |      |         |        |
| 6     | 149 |     |      |         |        |
| 7     | 17  |     |      |         |        |
| 8     | 61  |     |      |         |        |
| 9     | 60  |     |      |         |        |
| 10    | 124 |     |      |         |        |
| 11    | 96  |     |      |         |        |
| 12    | 75  |     |      |         |        |
| 13    | 310 |     |      |         |        |

페이지당 결과 수: 50 ▾ 1 ~ 50 (전체 3684행) | < > >>

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

- 상위 10개의 제품들을 출력하기

```

SELECT StockCode, COUNT(*) AS sell_cnt
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
limit 10;

```

project\_mainquest\_03

```

1 SELECT StockCode, COUNT(*) AS sell_cnt
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
3 GROUP BY StockCode
4 ORDER BY sell_cnt DESC
5 limit 10;

```

주문형 처리 할당량 사용 중

쿼리 결과

| 작업 정보 | 결과        | 시각화      | 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 수인지 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 `project-dac3b643-036f-483e-a72.modulabs_project.data`)
WHERE number_count <= 1;

```

project\_mainquest\_03

```

1 SELECT DISTINCT StockCode, number_count
2 FROM (SELECT StockCode,
3      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
4      FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`)
5 WHERE number_count <= 1;

```

주문형 처리 할당량 사용 중

쿼리 결과

| 작업 정보 | 결과           | 시각화          | JSON | 실행 세부정보 | 실행 그래프 |
|-------|--------------|--------------|------|---------|--------|
|       | 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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```

WITH stockcode_digits as (SELECT StockCode,
                               LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
                          FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`)
SELECT ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END)/count(*) * 100, 2) as StockCode_per
FROM stockcode_digits

```

[결과 이미지를 넣어주세요]

- (이미 DELETE 함수를 사용해 이미지 첨부 불가..)
  - 답 : 0.48%

- 제품과 관련되지 않은 거래 기록을 제거하기

```

DELETE FROM `project-dac3b643-036f-483e-a72.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 `project-dac3b643-036f-483e-a72.modulabs_project.data`) AS A
  WHERE A.number_count <= 1);

```

project\_mainquest\_03

```

1 DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
2 WHERE StockCode IN (
3   SELECT DISTINCT StockCode
4   FROM (SELECT StockCode,
5             LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
6           FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`) AS A
7   WHERE A.number_count <= 1);

```

쿼리 완료됨  
주문형 처리 할당량 사용 중

쿼리 결과

| 작업 정보 | 결과                                    | 실행 세부정보 | 실행 그래프 |
|-------|---------------------------------------|---------|--------|
|       | <b>이 문으로 data의 행 1,915개가 삭제되었습니다.</b> |         |        |

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```

SELECT Description, COUNT(*) AS description_cnt
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt desc
limit 30

```

project\_mainquest\_03

```

1 SELECT description, COUNT(*) AS description_cnt
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
3 GROUP BY Description
4 ORDER BY description_cnt desc
5 limit 30;

```

쿼리 완료됨

쿼리 결과

| 작업 정보 | 결과                             | 시각화             | JSON | 설명 세부정보 | 설명 그래프 |
|-------|--------------------------------|-----------------|------|---------|--------|
| 행     | Description                    | description_cnt |      |         |        |
| 1     | WHITE HANGING HEART TIER...    | 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 D... | 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 SPACEBODY DESIGN     | 1006            |      |         |        |
| 13    | LUNCH BAG CARS BLUE            | 1000            |      |         |        |

페이지당 결과 수: 50 | 1 ~ 30 (전체 308행) | < > >>

- 서비스 관련 정보를 포함하는 행들을 제거하기

```

DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE REGEXP_CONTAINS(UPPER>Description),
      r'(POSTAGE|CARRIAGE|AMAZON|BANK CHARGES|SAMPLES|TEST|MISSING|DAMAGED|ADJUST|DOTCOMPOSTAGE)');

```

```

1 DELETE FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
2 WHERE REGEXP_CONTAINS(UPPER(`Description`),
3   r'([A-Z].*[a-z])|([a-z].*[A-Z])');

```

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

이 문으로 data의 행 146개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```

CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
CASE
  WHEN REGEXP_CONTAINS(`Description`, r'([A-Z].*[a-z])|([a-z].*[A-Z])')
    THEN UPPER(`Description`)
  ELSE `Description`
END AS `Description`
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;

```

```

1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` AS
2 SELECT
3   * EXCEPT (`Description`),
4 CASE
5   WHEN REGEXP_CONTAINS(`Description`, r'([A-Z].*[a-z])|([a-z].*[A-Z])')
6     THEN UPPER(`Description`)
7   ELSE `Description`
8 END AS `Description`
9 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;

```

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다.

## 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 `project-dac3b643-036f-483e-a72.modulabs_project.data`;

```

project\_mainquest\_03

```
1 SELECT min(unitprice) AS min_price, max(unitprice) AS max_price, avg(unitprice) AS avg_price
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;
```

쿼리 완료됨

쿼리 결과

| 작업 정보 | 결과  | 시각화       | JSON              | 실행 세부정보   | 실행 그래프    |   |     |       |                   |  |  |  |  |
|-------|---|-----------|-------------------|-----------|-----------|---|-----|-------|-------------------|--|--|--|--|
|       | <table border="1"> <thead> <tr> <th>행</th> <th>min_price</th> <th>max_price</th> <th>avg_price</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0.0</td> <td>649.5</td> <td>2.910256885340...</td> </tr> </tbody> </table> | 행         | min_price         | max_price | avg_price | 1 | 0.0 | 649.5 | 2.910256885340... |  |  |  |  |
| 행     | min_price   | max_price | avg_price         |           |           |   |     |       |                   |  |  |  |  |
| 1     | 0.0   | 649.5     | 2.910256885340... |           |           |   |     |       |                   |  |  |  |  |

- 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 avg_quantity
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE unitprice = 0
```

project\_mainquest\_03

```
1 SELECT count(quantity) AS cnt_quantity, min(quantity) AS min_quantity, max(quantity) AS max_quantity, avg(quantity) AS avg_quantity
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
3 WHERE unitprice = 0
```

주문형 처리 할당량 사용 중

쿼리 결과

| 작업 정보 | 결과   | 시각화          | JSON         | 실행 세부정보           | 실행 그래프       |              |   |    |   |       |                   |  |  |  |  |
|-------|--|--------------|--------------|-------------------|--------------|--------------|---|----|---|-------|-------------------|--|--|--|--|
|       | <table border="1"> <thead> <tr> <th>행</th> <th>cnt_quantity</th> <th>min_quantity</th> <th>max_quantity</th> <th>avg.quantity</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>33</td> <td>1</td> <td>12540</td> <td>420.5151515151...</td> </tr> </tbody> </table> | 행            | cnt_quantity | min_quantity      | max_quantity | avg.quantity | 1 | 33 | 1 | 12540 | 420.5151515151... |  |  |  |  |
| 행     | cnt_quantity   | min_quantity | max_quantity | avg.quantity      |              |              |   |    |   |       |                   |  |  |  |  |
| 1     | 33   | 1            | 12540        | 420.5151515151... |              |              |   |    |   |       |                   |  |  |  |  |

- UnitPrice = 0** 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` AS
SELECT *
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
WHERE unitprice != 0
```

project\_mainquest\_03

```

1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.data` AS
2 SELECT *
3 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
4 WHERE unitprice != 0

```

쿼리 완료됨

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보    결과    실행 세부정보    실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다.

## 11-7. RFM 스코어

### Recency

- InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT *, cast(invoicedate as date) as InvoiceDay
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;
```

제목 있는 쿼리

```

1 SELECT *, cast(invoicedate as date) as InvoiceDay
2 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`;

```

주문형 처리 할당량 사용 중

쿼리 결과

| 작업 정보  | 결과      | 시작됨    | JSON                    | 설명 세부정보 | 설명 그림판 | InvoiceDay     |  |
|--------|---------|--------|-------------------------|---------|--------|----------------|--|
| 41451  | 23166   | 74015  | 2011-01-16 10:01:00 UTC | 1.04    | 12346  | United Kingdom | MEDIUM CERAMIC TOP STO... 2011-01-16       |
| 561433 | 23166   | -74215 | 2011-01-16 10:17:00 UTC | 1.04    | 12346  | United Kingdom | MEDIUM CERAMIC TOP STO... 2011-01-16       |
| 27626  | 22773   | 12     | 2010-12-07 14:57:00 UTC | 1.25    | 12347  | Ireland        | GREEN DRAWER KNOB ACRYL... 2010-12-07      |
| 27626  | 21171   | 12     | 2010-12-07 14:57:00 UTC | 1.45    | 12347  | Ireland        | BATHROOM METAL SIGN 2010-12-07             |
| 27626  | 20760   | 12     | 2010-12-07 14:57:00 UTC | 4.65    | 12347  | Ireland        | BLACK EAR MUFF FOR WOMENES 2010-12-07      |
| 27626  | 22779   | 4      | 2010-12-07 14:57:00 UTC | 9.75    | 12347  | Ireland        | ALARM CLOCK BATTERY OPERATED 2010-12-07    |
| 27626  | 22780   | 2      | 2010-12-07 14:57:00 UTC | 4.49    | 12347  | Ireland        | 5 X 15 RECORDED STACKING... 2010-12-07     |
| 27626  | 840598A | 24     | 2010-12-07 14:57:00 UTC | 2.95    | 12347  | Ireland        | 30 DOG PICTURE IN PLAYER CAR... 2010-12-07 |
| 27626  | 20792   | 4      | 2010-12-07 14:57:00 UTC | 5.49    | 12347  | Ireland        | CAMOUFLAGE EAR MUFF HEAD... 2010-12-07     |
| 27626  | 89116   | 12     | 2010-12-07 14:57:00 UTC | 2.1     | 12347  | Ireland        | BLACK CANDLABRA LIGHT 2010-12-07           |
| 27626  | 22728   | 4      | 2010-12-07 14:57:00 UTC | 3.75    | 12347  | Ireland        | ALARM CLOCK BATTERY PINN 2010-12-07        |
| 27626  | 22694   | 12     | 2010-12-07 14:57:00 UTC | 1.25    | 12347  | Ireland        | EMERGENCY FIRST AID TIN 2010-12-07         |

페이지당 결과 수: 50 | 1 - 50 (전체 398277건) | < > |< >|

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
WITH invo as (select *, cast(invoicedate as date) as invoiceday
      from `project-dac3b643-036f-483e-a72.modulabs_project.data`)
select max(invoiceday) as most_recent_date
from invo
```

④ project\_mainquest\_03

```
1 WITH invo AS (SELECT *, CAST(invoicedate AS DATE) AS invoiceday
2 | | | | | FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`)
3 SELECT MAX(invoiceday) AS most_recent_date
4 FROM invo
```

쿼리 완료됨

쿼리 결과

| 작업 정보 | 결과 | 시각화              | JSON | 실행 세부정보 | 실행 그래프 |
|-------|----|------------------|------|---------|--------|
|       | 행  | most_recent_date |      |         |        |
|       | 1  | 2011-12-09       |      |         |        |

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
select CustomerID,
       cast(invoicedate as date) as invoiceday,
       LAST_VALUE(invoicedate) over (partition by CustomerID order by invoicedate rows between current row and unbounded following) as last_invoiceday
  from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

⑤ project\_mainquest\_03

```
1 select CustomerID,
2       cast(invoicedate as date) as invoiceday,
3       LAST_VALUE(invoicedate) over (partition by CustomerID order by invoicedate rows between current row and unbounded following) as last_invoiceday
4   from `project-dac3b643-036f-483e-a72.modulabs_project.data`
```

쿼리 완료됨

주문항 처리 할당방 사용 중

쿼리 결과

| 작업 정보 | 결과 | 시각화        | JSON       | 실행 세부정보                 | 실행 그래프 |
|-------|----|------------|------------|-------------------------|--------|
|       | 행  | CustomerID | invoiceday | last_invoiceday         |        |
|       | 1  | 12346      | 2011-01-18 | 2011-01-18 10:17:00 UTC |        |
|       | 2  | 12346      | 2011-01-18 | 2011-01-18 10:17:00 UTC |        |
|       | 3  | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 4  | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 5  | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 6  | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 7  | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 8  | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 9  | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 10 | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 11 | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 12 | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | 13 | 12347      | 2010-10-07 | 2011-12-07 15:52:00 UTC |        |
|       | ** | **         | **         | **                      |        |

페이지당 결과 수: 50 ▾ 1 - 50 (전체 3982778개) | < > >>

- 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
);
```

제목 없는 쿼리

**실행** 저장 ▾ 다운로드 공유 ▾

```

1 SELECT
2   CustomerID,
3   EXTRACT(DAY FROM MAX(InvoiceDay)) OVER () - InvoiceDay) AS recency
4 FROM (
5   SELECT
6     CustomerID,
7     MAX(DATE(InvoiceDate)) AS InvoiceDay
8   FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
9   GROUP BY CustomerID
10 );

```

실행 시 이 쿼리가 6.08MB를 처리합니다.

주문형 처리 할당량 사용 중

### 쿼리 결과

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그레프

| 행  | CustomerID | recency |
|----|------------|---------|
| 1  | 12354      | 232     |
| 2  | 12388      | 15      |
| 3  | 12452      | 16      |
| 4  | 12633      | 58      |
| 5  | 12688      | 113     |
| 6  | 12830      | 37      |
| 7  | 12849      | 31      |
| 8  | 13426      | 0       |
| 9  | 13492      | 136     |
| 10 | 13507      | 3       |
| 11 | 13727      | 28      |
| 12 | 13953      | 7       |
| 13 | 14218      | 42      |

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

```

CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_r` AS
WITH user_r as (SELECT CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay)) OVER () - InvoiceDay) AS recency
FROM (SELECT CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`'
GROUP BY CustomerID))
select *
from user_r;

```

제목 없는 쿼리

**실행** 저장 ▾ 다운로드 공유 ▾ 일정 다음에서 열기 ▾ 더보기 ▾

```

1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_r` AS
2 WITH user_r as (SELECT CustomerID,
3   EXTRACT(DAY FROM MAX(InvoiceDay)) OVER () - InvoiceDay) AS recency
4   FROM (SELECT CustomerID,
5     MAX(DATE(InvoiceDate)) AS InvoiceDay
6   FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`'
7   GROUP BY CustomerID))
8 select *
9 from user_r;

```

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그레프

| 행 | CustomerID | recency |
|---|------------|---------|
| 1 | 12609      | 78      |
| 2 | 12716      | 3       |
| 3 | 12744      | 56      |
| 4 | 12906      | 11      |
| 5 | 12956      | 306     |
| 6 | 13130      | 94      |
| 7 | 13261      | 268     |
| 8 | 13270      | 366     |
| 9 | 13599      | 1       |

페이지당 결과 수: 50 ▾ 1 ~ 50 (전체 4361행) | < > >>

```

1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_r` AS
2 WITH user_r as (SELECT CustomerID,
3 EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
4 FROM (SELECT CustomerID,
5 MAX(DATE(InvoiceDate)) AS InvoiceDay
6 FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
7 GROUP BY CustomerID)
8 select *
9 from user_r;

```

● 실행 시 이 쿼리가 6.08MB를 처리합니다.  
주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

● 이 문으로 이름이 user\_r인 테이블이 교체되었습니다.

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
select customerID, count(distinct invoiceNo) as purchase_cnt
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by customerID
```

| customerID | purchase_cnt |
|------------|--------------|
| 1          | 12346        |
| 2          | 12347        |
| 3          | 12348        |
| 4          | 12349        |
| 5          | 12350        |
| 6          | 12352        |
| 7          | 12353        |
| 8          | 12354        |
| 9          | 12355        |
| 10         | 12356        |
| 11         | 12357        |
| 12         | 12358        |
| 13         | 12359        |
| 14         | 12360        |
| 15         | 12361        |
| 16         | 12362        |
| 17         | 12363        |

페이지당 결과 수: 50 | 1 - 50 (전체 4361행) | < > >>

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
select customerID, sum(Quantity) as item_cnt
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by customerID
```

제목 없는 쿼리

```
1 select customerID, sum(Quantity) as item_cnt
2 from `project-dac3b643-036f-483e-a72.modulabs_project.data`
3 group by customerID
```

쿼리 완료됨

쿼리 결과

| 작업 정보 | 결과         | 시각화      | JSON | 실행 세부정보 | 실행 그래프 |
|-------|------------|----------|------|---------|--------|
|       |            |          |      |         |        |
| 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     |      |         |        |
| 11    | 12357      | 2708     |      |         |        |
| 12    | 12358      | 242      |      |         |        |
| 13    | 12359      | 1599     |      |         |        |
| 14    | 12360      | 1156     |      |         |        |
| 15    | 12361      | 90       |      |         |        |
| 16    | 12362      | 2168     |      |         |        |
| 17    | 12363      | 408      |      |         |        |

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

```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
select customerID, count(distinct invoiceNo) as purchase_cnt
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by customerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
select customerID, sum(Quantity) as item_cnt
from `project-dac3b643-036f-483e-a72.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 `project-dac3b643-036f-483e-a72.modulabs_project.user_r` AS ur
ON pc.CustomerID = ur.CustomerID;

```

The screenshot shows a database query editor interface. At the top, there are tabs for '실행' (Run), '쿼리 저장' (Save Query), '다운로드' (Download), '공유' (Share), and '일정' (Schedule). Below the tabs is the query code:

```

1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_rf` AS
2
3 -- (1) 전체 거래 건수 계산
4 WITH purchase_cnt AS (
5   select customerID, count(distinct invoiceNo) as purchase_cnt
6   from `project-dac3b643-036f-483e-a72.modulabs_project.data`
7   group by customerID
8 )
9

```

Below the code, a message says '주문형 처리 할당량 사용 중' (Order-type processing allocation in use).

Under the heading '쿼리 결과' (Query Results), there are tabs for '작업 정보' (Job Info), '결과' (Results), '실행 세부정보' (Execution Details), and '실행 그래프' (Execution Graph). The '결과' tab is selected, showing a message: '이 문으로 이들이 user\_rf인 테이블이 교체되었습니다.' (This statement has replaced the user\_rf table).

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

select CustomerID, round(sum(Quantity * UnitPrice), 0) as user_total
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by CustomerID
having user_total != 0

```

The screenshot shows the results of the query in a table format. The table has columns: '작업 정보' (Job Info), '결과' (Results), '시각화' (Visualization), 'JSON' (JSON), '실행 세부정보' (Execution Details), and '실행 그래프' (Execution Graph). The '결과' tab is selected, displaying the following data:

| CustomerID | user_total |
|------------|------------|
| 12347      | 4910.0     |
| 12348      | 1437.0     |
| 12349      | 1458.0     |
| 12350      | 294.0      |
| 12352      | 1265.0     |
| 12353      | 89.0       |
| 12354      | 1079.0     |
| 12355      | 459.0      |
| 12356      | 2487.0     |
| 12357      | 6208.0     |
| 12358      | 928.0      |
| 12359      | 6183.0     |
| 12360      | 2302.0     |
| 12361      | 175.0      |
| 12362      | 4655.0     |

At the bottom right, it says '페이지당 결과 수: 50' (Results per page: 50) and '1 - 50 (전체 4348명)' (1 - 50 (Total 4348 people)).

- 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 `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm` AS
SELECT
rf.CustomerID AS CustomerID,
rf.purchase_cnt,
rf.item_cnt,

```

```

rf.reency,
ut.user_total,
round((ut.user_total/rf.purchase_cnt), 0) as user_average
FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_rf` as rf
LEFT JOIN (
select CustomerID, round(sum(Quantity * UnitPrice), 0) as user_total
from `project-dac3b643-036f-483e-a72.modulabs_project.data`
group by CustomerID
having user_total != 0) as ut
ON rf.CustomerID = ut.CustomerID;

```

The screenshot shows a database interface with a query editor and a results pane.

**Query Editor:**

```

1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm` AS
2 SELECT
3   rf.CustomerID AS CustomerID,
4   rf.purchase_cnt,
5   rf.item_cnt,
6   rf.reency,
7   ut.user_total,
8   round((ut.user_total/rf.purchase_cnt), 0) AS user_average
9   FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_rf` AS rf
10  LEFT JOIN (

```

**Results Pane:**

작업 정보    결과    실행 세부정보    실행 그래프

이 문으로 이름이 user\_rfm인 테이블이 교체되었습니다.

## RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```

select *
from `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm`

```

The screenshot shows a database interface with a query editor and a results pane.

**Query Editor:**

```

1 select *
2 from `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm`

```

**Results Pane:**

| Index | CustomerID | purchase_cnt | item_cnt | reency | user_total | user_average |
|-------|------------|--------------|----------|--------|------------|--------------|
| 1     | 12713      | 1            | 505      | 0      | 795.0      | 795.0        |
| 2     | 14569      | 1            | 79       | 1      | 227.0      | 227.0        |
| 3     | 15520      | 1            | 314      | 1      | 343.0      | 343.0        |
| 4     | 13436      | 1            | 76       | 1      | 197.0      | 197.0        |
| 5     | 13298      | 1            | 96       | 1      | 360.0      | 360.0        |
| 6     | 15195      | 1            | 1404     | 2      | 3861.0     | 3861.0       |
| 7     | 14024      | 1            | 72       | 2      | 151.0      | 151.0        |
| 8     | 15471      | 1            | 255      | 2      | 448.0      | 448.0        |
| 9     | 15992      | 1            | 17       | 3      | 42.0       | 42.0         |
| 10    | 14578      | 1            | 240      | 3      | 169.0      | 169.0        |
| 11    | 12478      | 1            | 233      | 3      | 546.0      | 546.0        |
| 12    | 17914      | 1            | 457      | 3      | 329.0      | 329.0        |
| 13    | 12650      | 1            | 250      | 3      | 242.0      | 242.0        |

페이지당 결과 수: 50 1 - 50 (전체 4362행) ⏪ ⏴ ⏵ ⏶ ⏷ ⏸

## 11-8. 추가 Feature 추출

### 1. 구매하는 제품의 다양성

- 1) 고객 별로 구매한 상품들의 고유한 수를 계산하기
- 2) `user_rfm` 테이블과 결과를 합치기
- 3) `user_data`라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

❶ 이 문으로 이름이 user\_data인 새 테이블이 생성되었습니다.

## 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-dac3b643-036f-483e-a72.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-dac3b643-036f-483e-a72.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
```

```

GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

```

```

1 CREATE OR REPLACE TABLE `project-dac3b643-036f-483e-a72.modulabs_project.user_data` AS
2 WITH purchase_intervals AS (
3   -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
4   SELECT
5     CustomerID,
6     CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
7   FROM (
8     -- (1) 구매와 구매 사이에 소요된 일수
9     SELECT
10       CustomerID,
11       DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
12     FROM
13     | `project-dac3b643-036f-483e-a72.modulabs_project.data`
14     WHERE CustomerID IS NOT NULL
15   )
16 GROUP BY CustomerID
17 )

```

쿼리 완료됨  
주문형 처리 할당량 사용 중

작업 정보    결과    실행 세부정보    실행 그래프

이 문으로서 이름이 user\_data인 테이블이 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 `project-dac3b643-036f-483e-a72.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT invoiceno) AS total_transactions,
    COUNT(DISTINCT CASE WHEN invoiceno LIKE 'C%' THEN invoiceno END) AS cancel_frequency
  FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
  WHERE customerID IS NOT NULL
  GROUP BY customerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency * 1.0 / t.total_transactions, 2) as cancel_rate
FROM `project-dac3b643-036f-483e-a72.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
  ON u.CustomerID = t.CustomerID;

```

```

5   CustomerID,
6   COUNT(DISTINCT invoiceNo) AS total_transactions,
7   COUNT(DISTINCT CASE WHEN invoiceNo LIKE '%-%' THEN invoiceNo END) AS cancel_frequency
8   FROM `project-dac3b643-036f-483e-a72.modulabs_project.data`
9   WHERE customerID IS NOT NULL
10  GROUP BY customerID
11
12
13  SELECT u.* t.* EXCEPT(CustomerID) ROUND(t.cancel_frequency * 1.0 / t.total_transactions, 2) AS cancel_rate

```

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

❶ 이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

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

```
select *
from `project-dac3b643-036f-483e-a72.modulabs_project.user_data`
```

제목 없는 쿼리

```

1 select *
2 from `project-dac3b643-036f-483e-a72.modulabs_project.user_data`

```

실행 시 이 뷰가 974.66KB를 처리합니다.

주문형 처리 할당량 사용 중

쿼리 결과

| 작업 정보 | 결과  | 시각화 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|---|-----|------|---------|--------|
| 1     | CustomerID, purchase_cnt, item_cnt, recency, user_total, user_average, unique_products, average_interval, total_transactions, cancel_frequency, cancel_rate |     |      |         |        |
| 2     | 15944, 1, 50, 276, 325.0, 325.0, 6, 0.0, 1, 0, 0.0  |     |      |         |        |
| 3     | 17647, 1, 30, 65, 133.0, 133.0, 7, 0.0, 1, 0, 0.0   |     |      |         |        |
| 4     | 14699, 1, 77, 214, 109.0, 109.0, 8, 0.0, 1, 0, 0.0  |     |      |         |        |
| 5     | 16849, 1, 33, 186, 125.0, 125.0, 8, 0.0, 1, 0, 0.0  |     |      |         |        |
| 6     | 15447, 1, 85, 330, 155.0, 155.0, 9, 0.0, 1, 0, 0.0  |     |      |         |        |
| 7     | 15663, 1, 106, 196, 138.0, 138.0, 9, 0.0, 1, 0, 0.0   |     |      |         |        |
| 8     | 13581, 1, 30, 309, 118.0, 118.0, 10, 0.0, 1, 0, 0.0   |     |      |         |        |
| 9     | 17205, 1, 884, 53, 384.0, 384.0, 13, 0.0, 1, 0, 0.0   |     |      |         |        |
| 10    | 15143, 1, 150, 225, 259.0, 259.0, 14, 0.0, 1, 0, 0.0  |     |      |         |        |
| 11    | 12532, 1, 151, 30, 314.0, 314.0, 15, 0.0, 1, 0, 0.0   |     |      |         |        |
| 12    | 13127, 1, 127, 77, 259.0, 259.0, 15, 0.0, 1, 0, 0.0   |     |      |         |        |
| 13    | 12614, 1, 76, 277, 316.0, 316.0, 20, 0.0, 1, 0, 0.0   |     |      |         |        |
| 14    | 17985, 1, 171, 22, 631.0, 631.0, 20, 0.0, 1, 0, 0.0   |     |      |         |        |
| 15    | 17194, 1, 112, 360, 227.0, 227.0, 25, 0.0, 1, 0, 0.0  |     |      |         |        |
| 16    | 12729, 1, 231, 113, 417.0, 417.0, 25, 0.0, 1, 0, 0.0  |     |      |         |        |
|       | 12920, 1, 128, 17, 163.0, 163.0, 27, 0.0, 1, 0, 0.0   |     |      |         |        |

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

## 회고

Keep : 최대한 직접 코드를 작성해보며 화면과 다른 값이 나왔을 때 직접 오류를 수정해보는 과정을 겪어보며 어떤 코드를 작성해야하는지 이해한 것 같다.

Problem : 익숙하지 않은 코드를 마주했을 때 집중력이 급자하되어서 시간을 소요한 부분이 있었던 것 같다.

Try : 최대한 기본기를 탄탄하게 쌓기 위해 더 많은 연습 문제를 풀어봐야 할 것 같다. 그리고 문제나 이후에 분석할 때 최대한 어떤 방향을 원하는지 오해 없이 이해하기 위해 문제 및 제안을 어떻게 해결해나갈지 생각하는 시간도 많이 가져야 할 것 같았다.

### ▼ 아카이브

X

X