

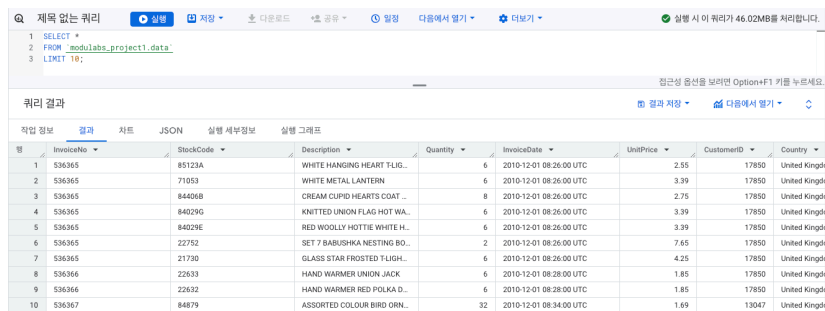
고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

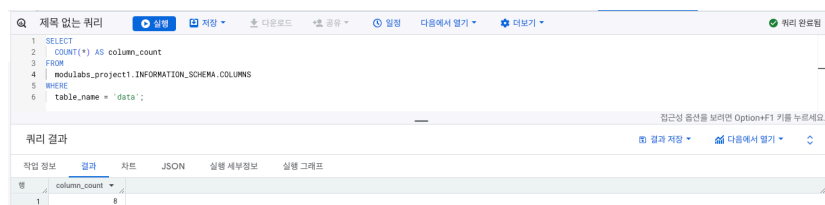
```
SELECT *  
FROM modulabs_project1.data  
LIMIT 10;
```



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART FLAG...	6	2010-12-01 08:29:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:29:00 UTC	3.39	17850	United Kingdom
3	536365	844069	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:29:00 UTC	2.75	17850	United Kingdom
4	536365	840296	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:29:00 UTC	3.39	17850	United Kingdom
5	536365	840296	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:29:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING RO...	2	2010-12-01 08:29:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T LIGH...	6	2010-12-01 08:29:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:29:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:29:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT  
COUNT(*) AS column_count  
FROM  
modulabs_project1.INFORMATION_SCHEMA.COLUMNS  
WHERE  
table_name = 'data';
```



행	column_count
1	8

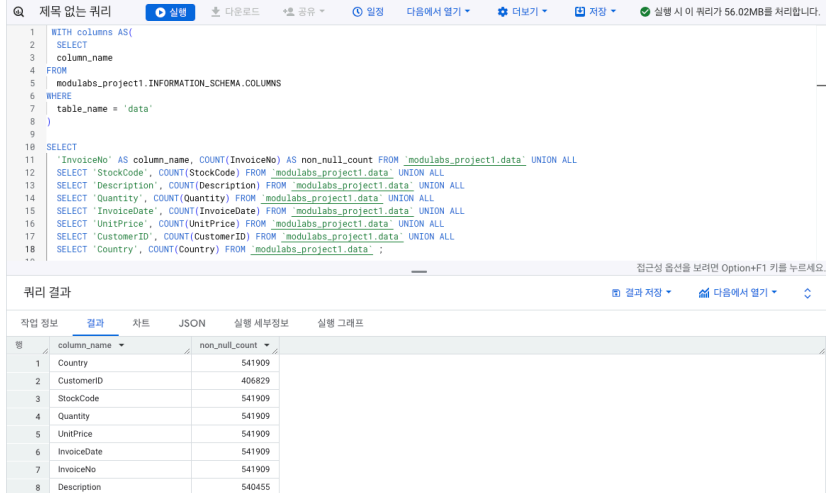
데이터 수 세기

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

```
WITH columns AS(  
SELECT  
column_name  
FROM  
modulabs_project1.INFORMATION_SCHEMA.COLUMNS  
WHERE  
table_name = 'data'  
)
```

SELECT

```
'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS non_null_count FROM `modulabs_project1.data` UNION ALL
SELECT 'StockCode', COUNT(StockCode) FROM `modulabs_project1.data` UNION ALL
SELECT 'Description', COUNT(Description) FROM `modulabs_project1.data` UNION ALL
SELECT 'Quantity', COUNT(Quantity) FROM `modulabs_project1.data` UNION ALL
SELECT 'InvoiceDate', COUNT(InvoiceDate) FROM `modulabs_project1.data` UNION ALL
SELECT 'UnitPrice', COUNT(UnitPrice) FROM `modulabs_project1.data` UNION ALL
SELECT 'CustomerID', COUNT(CustomerID) FROM `modulabs_project1.data` UNION ALL
SELECT 'Country', COUNT(Country) FROM `modulabs_project1.data` ;
```



쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	column_name	non_null_count			
1	Country	541909			
2	CustomerID	406829			
3	StockCode	541909			
4	Quantity	541909			
5	UnitPrice	541909			
6	InvoiceDate	541909			
7	InvoiceNo	541909			
8	Description	540455			

11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

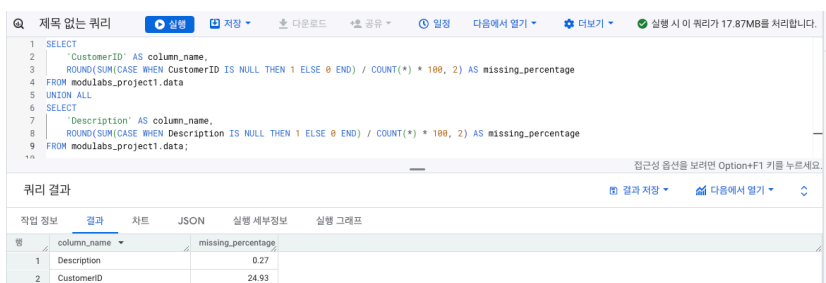
- 각 컬럼 별 누락된 값의 비율을 계산
 - 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

SELECT

```
'CustomerID' AS column_name,
ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM modulabs_project1.data
UNION ALL
```

SELECT

```
'Description' AS column_name,
ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM modulabs_project1.data;
```



쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	column_name	missing_percentage			
1	Description	0.27			
2	CustomerID	24.93			

결측치 처리 전략

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

```
SELECT Description, COUNT(Description)
FROM modulabs_project1.data
WHERE StockCode = '85123A'
GROUP BY Description;
```

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

```
1 SELECT Description, COUNT(Description)
2 FROM `modulabs_project1.data`
3 WHERE StockCode = '85123A'
4 GROUP BY Description;
```

Below the query, the results are displayed in a table:

행	Description	row_	row_
1	WHITE HANGING HEART T.LIG...		2302
2	?		1
3	wrongly marked carton 22804		1
4	CREAM HANGING HEART T.LIG...		9

결측치 처리

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

```
DELETE FROM `modulabs_project1.data`
WHERE CustomerID IS NULL;
```

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

```
1 /* CREATE TABLE `modulabs_project1.data_org` AS
2 SELECT *
3 FROM `modulabs_project1.data`;
4 */
5
6 -- #1 delete rows with null CustomerID
7 DELETE FROM `modulabs_project1.data`
8 WHERE CustomerID IS NULL;
9
```

Below the query, a message indicates the result of the deletion:

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, `COUNT`가 1보다 큰 데이터를 세어보기

```
SELECT *
FROM project_name.modulabs_project.data
(
  SELECT
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  FROM
    `modulabs_project1.data`
  GROUP BY
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING
```

[결과 : 4837행]

중복값 처리

- ```
CREATE TEMP TABLE temp_unique_table AS
SELECT DISTINCT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
FROM `modulabs_project1.data`;

DELETE FROM `modulabs_project1.data`
WHERE 1=1;

INSERT INTO `modulabs_project1.data`
SELECT *
FROM temp_unique_table;

SELECT COUNT(*)
FROM `modulabs_project1.data`;
```

[결과 401604행]

```
61
62 CREATE TEMP TABLE temp_unique_table AS
63 SELECT DISTINCT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
64 FROM `modulabs_project1.data`;
65
66 DELETE FROM `modulabs_project1.data`
67 WHERE 1=1;
68
69 INSERT INTO `modulabs_project1.data`
70 SELECT *
71 FROM temp_unique_table;
72
73 SELECT COUNT(*)
74 FROM `modulabs_project1.data`;
```

쿼리 결과

| 작업 정보 | 결과     | 차트 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|--------|----|------|---------|--------|
| 행     | f0_    |    |      |         |        |
| 1     | 401604 |    |      |         |        |

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM `modulabs_project1.data`;
```

[결과 22190행]

```
6 SELECT COUNT(DISTINCT InvoiceNo)
7 FROM `modulabs_project1.data`;
```

쿼리 결과

| 작업 정보 | 결과    | 차트 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|-------|----|------|---------|--------|
| 행     | f0_   |    |      |         |        |
| 1     | 22190 |    |      |         |        |

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT InvoiceNo
FROM `modulabs_project1.data`
GROUP BY InvoiceNo
LIMIT 100;
```

|    |                               |
|----|-------------------------------|
| 9  | SELECT InvoiceNo              |
| 10 | FROM `modulabs_project1.data` |
| 11 | GROUP BY InvoiceNo            |
| 12 | LIMIT 100                     |
| 13 |                               |

쿼리 결과

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

| 행  | 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 | 568172    |
| 14 | 577609    |
| 15 | 543037    |
| 16 | 544156    |
| 17 | 545323    |
| 18 | 545332    |
| 19 | 546869    |
| 20 | 547390    |
| 21 | 567395    |

페이지당 결과 수: 50 1 ~ 50 (전체 100행)

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

```
SELECT *
FROM `modulabs_project1.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

|    |                               |
|----|-------------------------------|
| 15 | SELECT *                      |
| 16 | FROM `modulabs_project1.data` |
| 17 | WHERE InvoiceNo LIKE 'C%'     |
| 18 | LIMIT 100                     |
| 19 |                               |
| 20 |                               |

쿼리 결과

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

| 행  | InvoiceNo | StockCode | Description                  | Quantity | InvoiceDate             | UnitPrice | CustomerID | Country        |
|----|-----------|-----------|------------------------------|----------|-------------------------|-----------|------------|----------------|
| 1  | C541433   | 23166     | MEDIUM CERAMIC TOP STOR..    | -74215   | 2011-01-18 10:17:00 UTC | 1.04      | 12346      | United Kingdom |
| 2  | C545329   | M         | Manual                       | -1       | 2011-03-01 15:47:00 UTC | 183.75    | 12352      | Norway         |
| 3  | C545329   | M         | Manual                       | -1       | 2011-03-01 15:47:00 UTC | 280.05    | 12352      | Norway         |
| 4  | C545330   | M         | Manual                       | -1       | 2011-03-01 15:49:00 UTC | 576.5     | 12352      | Norway         |
| 5  | C547388   | 21914     | BLUE HARMONICA IN BOX        | -12      | 2011-03-22 16:07:00 UTC | 1.25      | 12352      | Norway         |
| 6  | C547388   | 22701     | PINK DOG BOWL                | -6       | 2011-03-22 16:07:00 UTC | 2.95      | 12352      | Norway         |
| 7  | C547388   | 22784     | LANTERN CREAM GAZERO         | -3       | 2011-03-22 16:07:00 UTC | 4.95      | 12352      | Norway         |
| 8  | C547388   | 22413     | METAL SIGN TAKE IT OR LEAV.. | -6       | 2011-03-22 16:07:00 UTC | 2.95      | 12352      | Norway         |
| 9  | C547388   | 37448     | CERAMIC CAKE DESIGN SPOT..   | -12      | 2011-03-22 16:07:00 UTC | 1.49      | 12352      | Norway         |
| 10 | C547388   | 84050     | PINK HEART SHAPE EGG FRYL..  | -12      | 2011-03-22 16:07:00 UTC | 1.65      | 12352      | Norway         |
| 11 | C547388   | 22545     | CERAMIC HEART FAIRY CAKE ..  | -12      | 2011-03-22 16:07:00 UTC | 1.45      | 12352      | Norway         |
| 12 | C549955   | 22566     | RECIPE BOX PANTRY YELLOW ..  | -2       | 2011-04-13 13:38:00 UTC | 2.95      | 12359      | Cyprus         |
| 13 | C549955   | 22839     | 3 TIER CAKE TIN GREEN AND .. | -2       | 2011-04-13 13:38:00 UTC | 14.95     | 12359      | Cyprus         |
| 14 | C580165   | 22797     | CHEST OF DRAWERS GINDHAL ..  | -2       | 2011-12-02 11:21:00 UTC | 16.95     | 12359      | Cyprus         |
| 15 | C580165   | 22826     | LOVE SEAT ANTIQUE WHITE M..  | -1       | 2011-12-02 11:21:00 UTC | 42.5      | 12359      | Cyprus         |
| 16 | C580165   | 23245     | SET OF 3 REGENCY CAKE TING   | -2       | 2011-12-02 11:21:00 UTC | 4.95      | 12359      | Cyprus         |
| 17 | C580165   | 22720     | SET OF 3 CAKE TINS PANTRY .. | -1       | 2011-12-02 11:21:00 UTC | 4.95      | 12359      | Cyprus         |
| 18 | C544902   | 22273     | FELTCRAFT DOLL MOLLY         | -1       | 2011-02-24 13:05:00 UTC | 2.95      | 12362      | Belgium        |
| 19 | C544902   | 22629     | SPACEBOY LUNCH BOX           | -1       | 2011-02-24 13:05:00 UTC | 1.95      | 12362      | Belgium        |
| 20 | C563752   | 22659     | LUNCH BOX I LOVE LONDON      | -6       | 2011-08-19 10:38:00 UTC | 1.95      | 12362      | Belgium        |
| 21 | C563752   | 22991     | TEA FOR ONE POLKADOT         | -1       | 2011-08-19 10:38:00 UTC | 4.25      | 12362      | Belgium        |

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, 2) AS missing_percentage
FROM modulabs_project1.data;
```

[결과 2,21%]

|    |                                                                                                       |
|----|-------------------------------------------------------------------------------------------------------|
| 21 | SELECT                                                                                                |
| 22 | ROUND(SUM(CASE WHEN InvoiceNo Like 'C%' THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage |
| 23 | FROM modulabs_project1.data;                                                                          |
| 24 |                                                                                                       |
| 25 |                                                                                                       |

쿼리 결과

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

| 행 | missing_percentage |
|---|--------------------|
| 1 | 2.21               |

## StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM modulabs_project1.data;
```

[결과 3684행]

| 30                        |                                  |
|---------------------------|----------------------------------|
| 31                        | SELECT COUNT(DISTINCT StockCode) |
| 32                        | FROM modulabs_project1.data;     |
| 33                        |                                  |
| ...                       |                                  |
| 쿼리 결과                     |                                  |
| 작업 정보 <b>결과</b> 차트   JSON |                                  |
| 행                         | f0_                              |
| 1                         | 3684                             |

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

```
SELECT StockCode, COUNT(*) AS cnt
FROM modulabs_project1.data
GROUP BY StockCode
ORDER BY cnt DESC
LIMIT 10;
```

```

36 SELECT StockCode, COUNT(*) AS cnt
37 FROM modulabs_project1.data
38 GROUP BY StockCode
39 ORDER BY cnt DESC
40 LIMIT 10;
41

```

쿼리 결과

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

| 행  | StockCode | cnt  |
|----|-----------|------|
| 1  | 85123A    | 2065 |
| 2  | 22423     | 1894 |
| 3  | 85099B    | 1659 |
| 4  | 47566     | 1409 |
| 5  | 84879     | 1405 |
| 6  | 20725     | 1346 |
| 7  | 22720     | 1224 |
| 8  | POST      | 1196 |
| 9  | 22197     | 1110 |
| 10 | 23203     | 1108 |

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

```
SELECT DISTINCT StockCode
FROM `modulabs_project1.data`
```

```
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) = 0 OR
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) = 1;
```

[결과 8개]

```
84 SELECT DISTINCT StockCode
85 FROM `modulabs_project1.data`
86 WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) = 0 OR
87 LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) = 1;
```

쿼리 결과

| 작업 정보 | 결과           | 차트 | JSON | 실행 세부정보 | 실행 그래프 |
|-------|--------------|----|------|---------|--------|
| 행     | StockCode    |    |      |         |        |
| 1     | POST         |    |      |         |        |
| 2     | M            |    |      |         |        |
| 3     | C2           |    |      |         |        |
| 4     | D            |    |      |         |        |
| 5     | BANK CHARGES |    |      |         |        |
| 6     | PADS         |    |      |         |        |
| 7     | DOT          |    |      |         |        |
| 8     | CRUK         |    |      |         |        |

- **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
 'StockCode' AS column_name,
 ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM (
 SELECT StockCode,
 LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
 FROM `modulabs_project1.data`
);
```

[결과 0.48%]

```
101 SELECT
102 'StockCode' AS column_name,
103 ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
104 FROM (
105 SELECT StockCode,
106 LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
107 FROM `modulabs_project1.data`
108);
```

쿼리 결과

| 작업 정보 | 결과          | 차트 | JSON               | 실행 세부정보 | 실행 그래프 |
|-------|-------------|----|--------------------|---------|--------|
| 행     | column_name |    | missing_percentage |         |        |
| 1     | StockCode   |    | 0.48               |         |        |

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `modulabs_project1.data`
WHERE StockCode IN (
 SELECT DISTINCT StockCode
 FROM `modulabs_project1.data`
 WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1
);
```

[결과 1915행 삭제]



|     |                                                                               |
|-----|-------------------------------------------------------------------------------|
| 123 | DELETE FROM `modulabs_project1.data`                                          |
| 124 | WHERE StockCode IN (                                                          |
| 125 | SELECT DISTINCT StockCode                                                     |
| 126 | FROM `modulabs_project1.data`                                                 |
| 127 | WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1 |
| 128 | );                                                                            |

← 쿼리 결과

작업 정보   **결과**   실행 세부정보   실행 그래프

❗

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 `modulabs\_project1.data`  
GROUP BY Description  
ORDER BY Description\_cnt DESC  
LIMIT 30;

|     |                                                 |
|-----|-------------------------------------------------|
| 132 | SELECT Description, COUNT(*) AS description_cnt |
| 133 | FROM `modulabs_project1.data`                   |
| 134 | GROUP BY Description                            |
| 135 | ORDER BY Description_cnt DESC                   |
| 136 | LIMIT 30;                                       |

쿼리 결과

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

| 행  | Description                    | description_cnt |
|----|--------------------------------|-----------------|
| 1  | WHITE HANGING HEART TULG...    | 2058            |
| 2  | REGENCY CAKESTAND 3 TIER       | 1894            |
| 3  | JUMBO BAG RED RETROSPOT        | 1459            |
| 4  | PARTY BUNTING                  | 1409            |
| 5  | ASSORTED COLOUR BIRD ORN...    | 1405            |
| 6  | LUNCH BAG RED RETROSPOT        | 1345            |
| 7  | SET OF 3 CAKE TINS PANTRY ...  | 1224            |
| 8  | LUNCH BAG BLACK SKULL          | 1099            |
| 9  | PACK OF 72 RETROSPOT CAKE...   | 1062            |
| 10 | SPOTTY BUNTING                 | 1026            |
| 11 | PAPER CHAIN KIT 50'S CHRIST... | 1013            |
| 12 | LUNCH BAG SPACEBOY DESIGN      | 1006            |
| 13 | LUNCH BAG CARS BLUE            | 1000            |
| 14 | HEART OF WICKER SMALL          | 990             |
| 15 | NATURAL SLATE HEART CHAL...    | 989             |
| 16 | JAM MAKING SET WITH JARS       | 966             |
| 17 | LUNCH BAG PINK POLKADOT        | 961             |
| 18 | LUNCH BAG SUKI DESIGN          | 932             |
| 19 | ALARM CLOCK BAKELIKE RED       | 917             |
| 20 | REX CASH+CARRY JUMBO SH...     | 900             |

페이지당 결과 수: 50 ▼   1 ~ 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE FROM `modulabs\_project1.data`  
WHERE  
REGEXP\_CONTAINS(Description, r'High') -- High Resolution Image  
OR  
REGEXP\_CONTAINS(Description, r'Next') -- Next Day Carriag  
;

|     |                                                                |
|-----|----------------------------------------------------------------|
| 144 | DELETE FROM `modulabs_project1.data`                           |
| 145 | WHERE                                                          |
| 146 | REGEXP_CONTAINS(Description, r'High') -- High Resolution Image |
| 147 | OR                                                             |
| 148 | REGEXP_CONTAINS(Description, r'Next') -- Next Day Carriag      |
| 149 | ;                                                              |

쿼리 결과

작업 정보   **결과**   실행 세부정보   실행 그래프

❗

이 문으로 data의 행 837개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
UPDATE `modulabs_project1.data`
SET Description = UPPER(Description)
WHERE Description != UPPER(Description)
```

```
156 UPDATE `modulabs_project1.data`
157 SET Description = UPPER(Description)
158 WHERE Description != UPPER(Description)
159
```

← 쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

**i** 이 문으로 data의 행 1,296개가 수정되었습니다.

## UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS Minimum_p, MAX(UnitPrice) AS Maximum_p, AVG(UnitPrice)AS Average_p
FROM `modulabs_project1.data`;
```

```
169 SELECT MIN(UnitPrice) AS Minimum_p, MAX(UnitPrice) AS Maximum_p, AVG(UnitPrice)AS Average_p
170 FROM `modulabs_project1.data`;

```

쿼리 결과

| 작업 정보 | 결과          | 차트          | JSON               | 실행 세부정보 | 실행 그래프 |
|-------|-------------|-------------|--------------------|---------|--------|
| 행     | Minimum_p ▾ | Maximum_p ▾ | Average_p ▾        |         |        |
| 1     | 0.0         | 649.5       | 2.9049567574060506 |         |        |

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

```
SELECT COUNT(UnitPrice) AS Zero_p_cnt, MIN(Quantity) AS Minimum_q, MAX(Quantity) AS Maximum_q, AVG(Quantity)AS Average_q
FROM `modulabs_project1.data`
GROUP BY UnitPrice
HAVING UnitPrice = 0
```

```
171 SELECT COUNT(UnitPrice) AS Zero_p_cnt, MIN(Quantity) AS Minimum_q, MAX(Quantity) AS Maximum_q, AVG(Quantity)AS Average_q
172 FROM `modulabs_project1.data`
173 GROUP BY UnitPrice
174 HAVING UnitPrice = 0

```

쿼리 결과

| 작업 정보 | 결과           | 차트          | JSON        | 실행 세부정보           | 실행 그래프 |
|-------|--------------|-------------|-------------|-------------------|--------|
| 행     | Zero_p_cnt ▾ | Minimum_q ▾ | Maximum_q ▾ | Average_q ▾       |        |
| 1     | 33           | 1           | 12540       | 420.5151515151... |        |

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
DELETE FROM modulabs_project1.data
WHERE UnitPrice = 0;
```

[결과 33행 삭제됨]

```
177 DELETE FROM `modulabs_project1.data`
178 WHERE UnitPrice = 0;
179
```

쿼리 결과

작업 정보   **결과**   실행 세부정보   실행 그래프

**i** 이 문으로 data의 행 33개가 삭제되었습니다.

## 11-7. RFM 스코어

### Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT FORMAT_DATE("%Y/%m/%d", InvoiceDate) AS InvoiceDay, *
FROM `modulabs_project1.data`;
```

[결과]

| 일  | InvoiceDay | InvoiceNo | StockCode | Description                    | Quantity | InvoiceDate             | UnitPrice | CustomerID | Country |
|----|------------|-----------|-----------|--------------------------------|----------|-------------------------|-----------|------------|---------|
| 1  | 2011/01/18 | 541431    | 23166     | MEDIUM CERAMIC TOP STOR...     | 74215    | 2011-01-18 10:01:00 UTC | 1.04      | 12346      | United  |
| 2  | 2011/01/18 | CS41433   | 23166     | MEDIUM CERAMIC TOP STOR...     | -74215   | 2011-01-18 10:17:00 UTC | 1.04      | 12346      | United  |
| 3  | 2010/12/07 | 537626    | 84558A    | 3D DOG PICTURE PLAYING CA...   | 24       | 2010-12-07 14:57:00 UTC | 2.95      | 12347      | Iceland |
| 4  | 2010/12/07 | 537626    | 22487     | SET OF 2 TINS VINTAGE BATH...  | 4        | 2010-12-07 14:57:00 UTC | 4.25      | 12347      | Iceland |
| 5  | 2010/12/07 | 537626    | 22492     | MIM PAINT SET VINTAGE          | 36       | 2010-12-07 14:57:00 UTC | 6.65      | 12347      | Iceland |
| 6  | 2010/12/07 | 537626    | 22729     | ALARM CLOCK BAKELIKE ORA...    | 4        | 2010-12-07 14:57:00 UTC | 3.75      | 12347      | Iceland |
| 7  | 2010/12/07 | 537626    | 71477     | COLORFUL GLASS STAR TSLIGHT... | 12       | 2010-12-07 14:57:00 UTC | 3.25      | 12347      | Iceland |
| 8  | 2010/12/07 | 537626    | 21171     | BATHROOM METAL SIGN            | 12       | 2010-12-07 14:57:00 UTC | 1.45      | 12347      | Iceland |
| 9  | 2010/12/07 | 537626    | 20780     | BLACK EAR MUFF HEADPHON...     | 12       | 2010-12-07 14:57:00 UTC | 4.65      | 12347      | Iceland |
| 10 | 2010/12/07 | 537626    | 22775     | PURPLE DRAWERKNOB ACRYL...     | 12       | 2010-12-07 14:57:00 UTC | 1.25      | 12347      | Iceland |
| 11 | 2010/12/07 | 537626    | 22795     | LARGE HEART MEASURING SP...    | 12       | 2010-12-07 14:57:00 UTC | 1.65      | 12347      | Iceland |
| 12 | 2010/12/07 | 537626    | 849970    | PINK 5 PIECE POLAROID CUT...   | 6        | 2010-12-07 14:57:00 UTC | 3.75      | 12347      | Iceland |
| 13 | 2010/12/07 | 537626    | 21731     | RED TOASTDISH LED NIGHT L...   | 12       | 2010-12-07 14:57:00 UTC | 1.65      | 12347      | Iceland |
| 14 | 2010/12/07 | 537626    | 84969     | BOX OF 6 ASSORTED COLOUR ...   | 6        | 2010-12-07 14:57:00 UTC | 4.25      | 12347      | Iceland |
| 15 | 2010/12/07 | 537626    | 22725     | ALARM CLOCK BAKELIKE GRE...    | 4        | 2010-12-07 14:57:00 UTC | 3.75      | 12347      | Iceland |
| 16 | 2010/12/07 | 537626    | 851678    | BLACK GRAND BAROQUE PHO...     | 30       | 2010-12-07 14:57:00 UTC | 1.25      | 12347      | Iceland |
| 17 | 2010/12/07 | 537626    | 22773     | GREEN DRAWER KNOB ACRYL...     | 12       | 2010-12-07 14:57:00 UTC | 1.25      | 12347      | Iceland |
| 18 | 2010/12/07 | 537626    | 22725     | ALARM CLOCK BAKELIKE CHO...    | 4        | 2010-12-07 14:57:00 UTC | 3.75      | 12347      | Iceland |

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
 FORMAT_DATE("%Y/%m/%d", MAX(InvoiceDate)) AS most_recent_Invoice_date
FROM `modulabs_project1.data`
```

[결과 : 2011/12/09 ]



[결과]

```

33 -- 가장 최근 일자(most_recent_date)와 유지별 마지막 구매일(InvoiceDay)간의 차이를 계산하기
34 SELECT
35 CustomerID,
36 EXTRACT(DAY FROM MAX(recent_InvDay_CID) OVER () - recent_InvDay_CID) AS recency
37 FROM (
38 SELECT
39 CustomerID, MAX(DATE(InvoiceDate)) AS recent_InvDay_CID,
40 FROM `modulabs_project1.data`
41 GROUP BY CustomerID
42)
43 ORDER BY recency;
44

```

쿼리 결과

결과 저장 다음에서 열기

| 작업 정보 | 결과         | 차트      | JSON | 실행 세부정보 | 실행 그래프 |
|-------|------------|---------|------|---------|--------|
| 행     | CustomerID | recency |      |         |        |
| 1     | 13113      | 0       |      |         |        |
| 2     | 13777      | 0       |      |         |        |
| 3     | 16446      | 0       |      |         |        |
| 4     | 13426      | 0       |      |         |        |
| 5     | 16954      | 0       |      |         |        |
| 6     | 15910      | 0       |      |         |        |
| 7     | 17315      | 0       |      |         |        |
| 8     | 15804      | 0       |      |         |        |
| 9     | 15694      | 0       |      |         |        |
| 10    | 12433      | 0       |      |         |        |
| 11    | 14051      | 0       |      |         |        |
| 12    | 12748      | 0       |      |         |        |
| 13    | 14397      | 0       |      |         |        |
| 14    | 15311      | 0       |      |         |        |
| 15    | 15344      | 0       |      |         |        |
| 16    | 16558      | 0       |      |         |        |

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

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

```

CREATE OR REPLACE TABLE `modulabs_project1.user_r` AS
SELECT
 CustomerID,
 EXTRACT(DAY FROM MAX(recent_InvDay_CID) OVER () - recent_InvDay_CID) AS recency
FROM (
 SELECT
 CustomerID, MAX(DATE(InvoiceDate)) AS recent_InvDay_CID,
 FROM `modulabs_project1.data`
 GROUP BY CustomerID
)
ORDER BY recency;

```

[결과]

```

47 CREATE OR REPLACE TABLE `modulabs_project1.user_r` AS
48 SELECT
49 CustomerID,
50 EXTRACT(DAY FROM MAX(recent_InvDay_CID) OVER () - recent_InvDay_CID) AS recency
51 FROM (
52 SELECT
53 CustomerID, MAX(DATE(InvoiceDate)) AS recent_InvDay_CID,
54 FROM `modulabs_project1.data`
55 GROUP BY CustomerID
56)
57 ORDER BY recency;

```

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

**i** 이 문으로 이름이 user\_r인 새 테이블이 생성되었습니다.

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS order_cnt
FROM `modulabs_project1.data`
GROUP BY CustomerID;
```

[결과]

```
60 SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS order_cnt
61 FROM `modulabs_project1.data`
62 GROUP BY CustomerID;
```

접근성 옵션을 보러

← 쿼리 결과

결과 저장

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

| 행  | CustomerID | order_cnt |
|----|------------|-----------|
| 1  | 12346      | 2         |
| 2  | 12347      | 7         |
| 3  | 12348      | 4         |
| 4  | 12349      | 1         |
| 5  | 12350      | 1         |
| 6  | 12352      | 8         |
| 7  | 12353      | 1         |
| 8  | 12354      | 1         |
| 9  | 12355      | 1         |
| 10 | 12356      | 3         |
| 11 | 12357      | 1         |
| 12 | 12358      | 2         |
| 13 | 12359      | 6         |
| 14 | 12360      | 3         |
| 15 | 12361      | 1         |
| 16 | 12362      | 13        |
| 17 | 12363      | 2         |
| 18 | 12364      | 4         |
| 19 | 12365      | 1         |
| 20 | 12367      | 1         |

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT CustomerID, COUNT(InvoiceNo) AS item_cnt
FROM `modulabs_project1.data`
GROUP BY CustomerID;
```

[결과]

```

64 SELECT CustomerID, COUNT(InvoiceNo) AS item_cnt
65 FROM `modulabs_project1.data`
66 GROUP BY CustomerID;

```

접근성 옵션을 보

← 쿼리 결과 결과 저장 ▼

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

| 행  | CustomerID | item_cnt |
|----|------------|----------|
| 1  | 12346      | 2        |
| 2  | 12347      | 182      |
| 3  | 12348      | 27       |
| 4  | 12349      | 72       |
| 5  | 12350      | 16       |
| 6  | 12352      | 84       |
| 7  | 12353      | 4        |
| 8  | 12354      | 58       |
| 9  | 12355      | 13       |
| 10 | 12356      | 58       |
| 11 | 12357      | 131      |
| 12 | 12358      | 17       |
| 13 | 12359      | 251      |
| 14 | 12360      | 126      |
| 15 | 12361      | 9        |
| 16 | 12362      | 264      |
| 17 | 12363      | 23       |
| 18 | 12364      | 81       |
| 19 | 12365      | 21       |
| 20 | 12367      | 10       |
| 21 | 12370      | 165      |

페이지당 결과 수: 50 ▼   1 - 50 (전체 4362행)

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

```

CREATE OR REPLACE TABLE `modulabs_project1.user_rf` AS
WITH order_cnt AS (
 SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS order_cnt
 FROM `modulabs_project1.data`
 GROUP BY CustomerID
),
item_cnt AS (
 SELECT CustomerID, COUNT(InvoiceNo) AS item_cnt
 FROM `modulabs_project1.data`
 GROUP BY CustomerID
)
SELECT
 pc.CustomerID,
 pc.order_cnt,
 ic.item_cnt,
 ur.recency
FROM order_cnt AS pc
JOIN item_cnt AS ic
 ON pc.CustomerID = ic.CustomerID
JOIN `modulabs_project1.user_r` AS ur
 ON pc.CustomerID = ur.CustomerID;

```

[결과]

```

69 CREATE OR REPLACE TABLE `modulabs_project1.user_rf` AS
70 WITH order_cnt AS (
71 SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS order_cnt
72 FROM `modulabs_project1.data`
73 GROUP BY CustomerID
74),
75 item_cnt AS (
76 SELECT CustomerID, COUNT(InvoiceNo) AS item_cnt
77 FROM `modulabs_project1.data`
78 GROUP BY CustomerID
79)
80 SELECT
81 pc.CustomerID,
82 pc.order_cnt,
83 ic.item_cnt,
84 ur.recency
85 FROM order_cnt AS pc
86 JOIN item_cnt AS ic
87 ON pc.CustomerID = ic.CustomerID
88 JOIN `modulabs_project1.user_r` AS ur
89 ON pc.CustomerID = ur.CustomerID;
90

```

## 쿼리 결과

작업 정보   결과   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 user\_rf인 새 테이블이 생성되었습니다.

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
 CustomerID,
 ROUND(SUM(order_sum), 1) AS user_total
FROM (
 SELECT CustomerID, Quantity, UnitPrice, Quantity * UnitPrice AS order_sum
 FROM `modulabs_project1.data`
)
GROUP BY CustomerID;

```

[결과]



|     |                                                                           |
|-----|---------------------------------------------------------------------------|
| 100 | SELECT                                                                    |
| 101 | CustomerID,                                                               |
| 102 | ROUND(SUM(order_sum), 1) AS user_total                                    |
| 103 | FROM (                                                                    |
| 104 | SELECT CustomerID, Quantity, UnitPrice, Quantity * UnitPrice AS order_sum |
| 105 | FROM `modulabs_project1.data`                                             |
| 106 | -- WHERE (Quantity * UnitPrice) <= 0 OR (Quantity * UnitPrice) >= 1000    |
| 107 | )                                                                         |
| 108 | GROUP BY CustomerID                                                       |
| 109 | --ORDER BY user_total;                                                    |

[접근성 옵션을 보](#)

**쿼리 결과**
[결과 저장](#)

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

| 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      | 1265.4     |
| 7  | 12353      | 89.0       |
| 8  | 12354      | 1079.4     |
| 9  | 12355      | 459.4      |
| 10 | 12356      | 2487.4     |
| 11 | 12357      | 6207.7     |
| 12 | 12358      | 928.1      |
| 13 | 12359      | 6183.0     |
| 14 | 12360      | 2302.1     |
| 15 | 12361      | 174.9      |
| 16 | 12362      | 4665.6     |
| 17 | 12363      | 552.0      |

페이지당 결과 수: 50 ▼   1 - 50 (전체 4362행)

## • 고객별 평균 거래 금액 계산

- 고객별 평균 거래 금액을 구하기 위해 1) `data` 테이블을 `user_rf` 테이블과 조인(LEFT JOIN) 한 후, 2) `purchase_cnt` 로 나누어서 3) `user_rfm` 테이블로 저장하기

```
CREATE OR REPLACE TABLE `modulabs_project1.user_rfm` AS
SELECT
 rf.CustomerID AS CustomerID,
 rf.order_cnt,
 rf.item_cnt,
 rf.recency,
 ut.user_total,
 ROUND(ut.user_total/rf.order_cnt, 1) AS user_average
FROM `modulabs_project1.user_rf` rf
LEFT JOIN (
 SELECT
 CustomerID,
 ROUND(SUM(order_sum), 1) AS user_total
 FROM (
 SELECT CustomerID, Quantity, UnitPrice, Quantity * UnitPrice AS order_sum
 FROM `modulabs_project1.data`
)
 GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 테이블 생성됨]

```

111 CREATE OR REPLACE TABLE `modulabs_project1.user_rfm` AS
112 SELECT
113 rf.CustomerID AS CustomerID,
114 rf.order_cnt,
115 rf.item_cnt,
116 rf.recency,
117 ut.user_total,
118 ROUND(ut.user_total/rf.order_cnt, 1) AS user_average
119 FROM `modulabs_project1.user_rf` rf

```

## 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 `modulabs_project1.user_rfm`
LIMIT 5;

```

[결과 이미지]

```

134 SELECT *
135 FROM `modulabs_project1.user_rfm`
136 LIMIT 5;
137

```

## 쿼리 결과

| 작업 정보 | 결과           | 차트          | JSON       | 실행 세부정보   | 실행 그래프       |                |
|-------|--------------|-------------|------------|-----------|--------------|----------------|
| 행     | CustomerID ▾ | order_cnt ▾ | item_cnt ▾ | recency ▾ | user_total ▾ | user_average ▾ |
| 1     | 16138        | 1           | 1          | 368       | -8.0         | -8.0           |
| 2     | 17763        | 1           | 1          | 263       | 15.0         | 15.0           |
| 3     | 16765        | 1           | 1          | 294       | 34.0         | 34.0           |
| 4     | 15562        | 1           | 1          | 351       | 134.6        | 134.6          |
| 5     | 15488        | 1           | 1          | 92        | 76.3         | 76.3           |

## 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 `modulabs_project1.user_data` AS
WITH unique_products AS (
SELECT
 CustomerID,
 COUNT(DISTINCT StockCode) AS unique_products
FROM `modulabs_project1.data`

```

```

GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `modulabs_project1.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;

```

[결과 이미지]

```

140 CREATE OR REPLACE TABLE `modulabs_project1.user_data` AS
141 WITH unique_products AS (
142 SELECT
143 CustomerID,
144 COUNT(DISTINCT StockCode) AS unique_products
145 FROM `modulabs_project1.data`
146 GROUP BY CustomerID
147)
148 SELECT ur.*, up.* EXCEPT (CustomerID)
149 FROM `modulabs_project1.user_rfm` AS ur
150 JOIN unique_products AS up
151 ON ur.CustomerID = up.CustomerID;
152

```

## 쿼리 결과

작업 정보   **결과**   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 user\_data인 새 테이블이 생성되었습니다.

## 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 `modulabs_project1.user_data` AS
WITH purchase_intervals AS (
 SELECT
 CustomerID,
 CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
 FROM (
 SELECT
 CustomerID,
 DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
 FROM
 `modulabs_project1.data`
 WHERE CustomerID IS NOT NULL
)
 GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `modulabs_project1.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

```

[결과 이미지]

```

153 CREATE OR REPLACE TABLE `modulabs_project1.user_data` AS
154 WITH purchase_intervals AS (
155 SELECT
156 CustomerID,
157 CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
158 FROM (
159 SELECT
160 CustomerID,
161 DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
162 FROM
163 `modulabs_project1.data`
164 WHERE CustomerID IS NOT NULL
165)
166 GROUP BY CustomerID
167)
168 SELECT u.*, pi.* EXCEPT (CustomerID)
169 FROM `modulabs_project1.user_data` AS u
170 LEFT JOIN purchase_intervals AS pi
171 ON u.CustomerID = pi.CustomerID;

```

쿼리 결과

작업 정보   결과   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

### 3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기
  - 1) 취소 빈도(cancel\_frequency) : 고객 별로 취소한 거래의 총 횟수
  - 2) 취소 비율(cancel\_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
    - 취소 빈도와 취소 비율을 계산하고 그 결과를 **user\_data** 에 통합하기  
(취소 비율은 소수점 두번째 자리)

```

CREATE OR REPLACE TABLE `modulabs_project1.user_data` AS

WITH TransactionInfo AS (
 SELECT
 CustomerID,
 COUNT(DISTINCT InvoiceNo) AS total_transactions,
 SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) AS cancel_frequency
 FROM `modulabs_project1.data`
 GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), (t.cancel_frequency / t.total_transactions) AS cancel_rate
FROM `modulabs_project1.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;

```

[결과 이미지]

```

179 CREATE OR REPLACE TABLE `modulabs_project1.user_data` AS
180
181 WITH TransactionInfo AS (
182 SELECT
183 CustomerID,
184 COUNT(DISTINCT InvoiceNo) AS total_transactions,
185 SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) AS cancel_frequency
186 FROM `modulabs_project1.data`
187 GROUP BY CustomerID
188)
189
190 SELECT u.*, t.* EXCEPT(CustomerID), (t.cancel_frequency / t.total_transactions) AS cancel_rate
191 FROM `modulabs_project1.user_data` AS u
192 LEFT JOIN TransactionInfo AS t
193 ON u.CustomerID = t.CustomerID;
194
195
196

```

## 쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

**i** 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 `modulabs_project1.user_data`
LIMIT 5;

```

[결과 이미지]

```

200 SELECT *
201 FROM `modulabs_project1.user_data`
202 LIMIT 5;
203

```

접근성 옵션을 보려면 Option+F1 키를 누

## 쿼리 결과

결과 저장

다음에서 열기

| 작업 정보 |            | 결과        | 차트       | JSON    | 실행 세부정보    |              | 실행 그래프          |                  |                    |                  |             |
|-------|------------|-----------|----------|---------|------------|--------------|-----------------|------------------|--------------------|------------------|-------------|
| 행     | CustomerID | order_cnt | item_cnt | recency | user_total | user_average | unique_products | average_interval | total_transactions | cancel_frequency | cancel_rate |
| 1     | 15524      | 1         | 1        | 24      | 440.0      | 440.0        | 1               | 0.0              | 1                  | 0                | 0.0         |
| 2     | 17382      | 1         | 1        | 65      | 50.4       | 50.4         | 1               | 0.0              | 1                  | 0                | 0.0         |
| 3     | 16953      | 1         | 1        | 30      | 20.8       | 20.8         | 1               | 0.0              | 1                  | 0                | 0.0         |
| 4     | 13967      | 1         | 2        | 145     | 80.7       | 80.7         | 2               | 0.0              | 1                  | 0                | 0.0         |
| 5     | 13160      | 1         | 4        | 22      | 91.8       | 91.8         | 4               | 0.0              | 1                  | 0                | 0.0         |

페이지당 결과 수: 50 1 - 5 (전체 5행) |< < >

## 회고

[회고 내용을 작성해주세요]

Keep : 집중한다.

Problem : 시간 배분이 안됐다.

Try : 소요예상 시간을 예상하고 나머지 하자.