고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

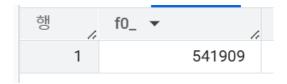
• 테이블에 있는 10개의 행만 출력하기

SELECT *
FROM bold-syntax-456104-m4.modulabs_project.data
LIMIT 10



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT COUNT(*)
FROM bold-syntax-456104-m4.modulabs_project.data



데이터 수 세기

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

SELECT
COUNT(invoiceNo) AS COUNT_invoiceNo,
COUNT(StockCode) AS COUNT_StockCode,
COUNT(Description) AS COUNT_Description,
COUNT(Quantity) AS COUNT_Quantity,
COUNT(InvoiceDate) AS COUNT_InvoiceDate,
COUNT(UnitPrice) AS COUNT_UnitPrice,
COUNT(CustomerId) AS COUNT_CustomerId,
COUNT(Country) AS COUNT_Country
FROM bold-syntax-456104-m4.modulabs_project.data;



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] UNION ALL

SELECT 'StockCode', ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) FROM `bold-sy UNION ALL

SELECT 'Description', ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) FROM 'bold-s' UNION ALL

SELECT 'Quantity', ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) FROM `bold-syntax-UNION ALL

SELECT 'InvoiceDate', ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) FROM `bold-s UNION ALL

SELECT 'UnitPrice', ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) FROM 'bold-synta' UNION ALL

SELECT 'CustomerID', ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) FROM `bold-t UNION ALL

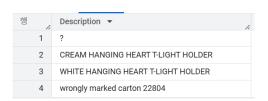
SELECT 'Country', ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) FROM 'bold-syntax-4

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

결측치 처리 전략

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

SELECT Description
FROM bold-syntax-456104-m4.modulabs_project.data
WHERE StockCode = '85123A'
GROUP BY Description
ORDER BY Description



결측치 처리

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

DELETE FROM bold-syntax-456104-m4.modulabs_project.data WHERE CustomerID IS NULL OR Description IS NULL

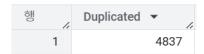
● 이 문으로 data의 행 135,080개가 삭제되었습니다.

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
FROM (
SELECT CustomerID, InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, Country, COUNT(*)
FROM bold-syntax-456104-m4.modulabs_project.data
GROUP BY CustomerID, InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, Country
HAVING COUNT(*) > 1
);
```



중복값 처리

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

CREATE OR REPLACE TABLE bold-syntax-456104-m4.modulabs_project.data AS
SELECT DISTINCT *
FROM bold-syntax-456104-m4.modulabs_project.data;

SELECT COUNT(*)
FROM bold-syntax-456104-m4.modulabs_project.data;





11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

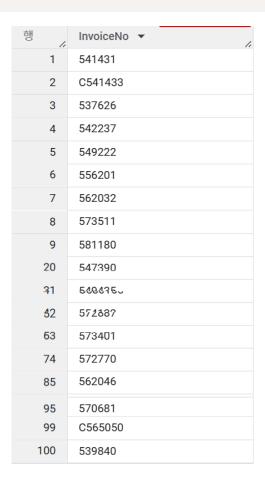
• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT COUNT(DISTINCT InvoiceNo)
FROM bold-syntax-456104-m4.modulabs_project.data;



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

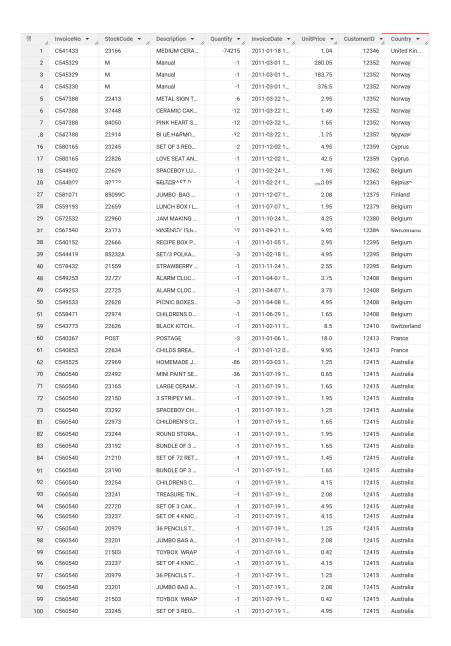
SELECT DISTINCT InvoiceNo FROM bold-syntax-456104-m4.modulabs_project.data LIMIT 100;



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

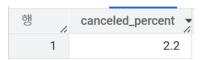
SELECT *

FROM bold-syntax-456104-m4.modulabs_project.data WHERE InvoiceNo LIKE 'C%' LIMIT 100;



• 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_ratio_percent
FROM bold-syntax-456104-m4.modulabs_project.data;
```



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT Stockcode)

FROM bold-syntax-456104-m4.modulabs_project.data;



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

SELECT Stockcode, COUNT(*) AS sell_cnt FROM bold-syntax-456104-m4.modulabs_project.data GROUP BY 1 ORDER BY sell_cnt DESC LIMIT 10

| 행 | 11 | Stockcode ▼ | sell_cnt | • | 1. |
|----|----|-------------|----------|---|------|
| 1 | | 85123A | | | 2065 |
| 2 | 2 | 22423 | | | 1894 |
| 3 | 3 | 85099B | | | 1659 |
| 4 | 1 | 47566 | | | 1409 |
| 5 | 5 | 84879 | | | 1405 |
| 6 | 5 | 20725 | | | 1346 |
| 7 | 7 | 22720 | | | 1224 |
| 8 | 3 | POST | | | 1196 |
| 9 | 9 | 22197 | | | 1110 |
| 10 |) | 23203 | | | 1108 |

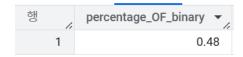
- StockCode 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 。 **숫자가 0~1개인 값**들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT StockCode, number_count
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM bold-syntax-456104-m4.modulabs_project.data
)
WHERE number_count BETWEEN 0 AND 1
```

| 행 // | 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)

```
SELECT ROUND(
COUNTIF(number_count BETWEEN 0 AND 1) / COUNT(*) * 100,
2
) AS percentage_OF_binary
FROM (
SELECT LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM bold-syntax-456104-m4.modulabs_project.data
);
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM bold-syntax-456104-m4.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 bold-syntax-456104-m4.modulabs_project.data
)

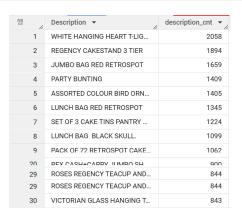
WHERE number_count BETWEEN 0 AND 1
);
```

● 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 bold-syntax-456104-m4.modulabs_project.data WHERE Description IS NOT NULL GROUP BY Description ORDER BY description_cnt DESC LIMIT 30;



• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM bold-syntax-456104-m4.modulabs_project.data
WHERE UPPER(Description) IN (
'NEXT DAY CARRIAGE',
'HIGH RESOLUTION IMAGE',
'POSTAGE',
'CARRIAGE',
'BANK CHARGES',
'DOT',
'MANUAL',
'M',
'SAMPLES',
'TEST'
);
```

❶ 이 문으로 data의 행 83개가 삭제되었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE bold-syntax-456104-m4.modulabs_project.data AS
SELECT

* EXCEPT (Description),
UPPER(Description) AS Description
FROM bold-syntax-456104-m4.modulabs_project.data;

● 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 bold-syntax-456104-m4.modulabs_project.data;



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



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE bold-syntax-456104-m4.modulabs_project.data AS SELECT *
FROM bold-syntax-456104-m4.modulabs_project.data
WHERE UnitPrice > 0;

● 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT

DATE(InvoiceDate) AS InvoiceDay,

*
FROM bold-syntax-456104-m4.modulabs_project.data;
```

| 행 // | InvoiceDay ▼ | InvoiceNo ▼ // | StockCode ▼ | Description ▼ // | Quantity 🕶 | InvoiceDate ▼// | UnitPrice 🔻 | CustomerID ▼ | Country 🔻 |
|------|--------------|----------------|-------------|------------------|------------|-----------------|--------------|--------------|----------------|
| 1 | 2011-01-18 | 541431 | 23166 | MEDIUM CER | 74215 | 2011-01-18 | 1.04 | 12346 | United Kingdom |
| 2 | 2011-01-18 | C541433 | 23166 | MEDIUM CER | -74215 | 2011-01-18 | 1.04 | 12346 | United Kingdom |
| 3 | 2010-12-07 | 537626 | 84558A | 3D DOG PICT | 24 | 2010-12-07 | 2.95 | 12347 | Iceland |
| 4 | 2010-12-07 | 537626 | 85232D | SET/3 DECO | 3 | 2010-12-07 | 4.95 | 12347 | Iceland |
| 5 | 2010-12-07 | 537626 | 22775 | PURPLE DRA | 12 | 2010-12-07 | 1.25 | 12347 | Iceland |
| 6 | 2010-12-07 | 537626 | 20780 | BLACK EAR | 12 | 2010-12-07 | 4.65 | 12347 | Iceland |
| 7 | 2010-12-07 | 537626 | 22771 | CLEAR DRAW | 12 | 2010-12-07 | 1.25 | 12347 | Iceland |
| 8 | 2010-12-07 | 537626 | 20782 | CAMOUFLAG | 6 | 2010-12-07 | 5.49 | 12347 | Iceland |
| 18 | 2010-12-07 | 537626 | 22272 | ^スᲐᲡᲮՐᲧᲪᲪ4 | ^6 | 2010-12-07 | <i>^∠</i> .ፕ | 12347 | Iceland |
| 17 | 2010-12-07 | 537626 | 22195 | LARGE HEAR | 12 | 2010-12-07 | 1.65 | 12347 | Iceland |
| 18 | 2010-12-07 | 537626 | 22497 | SET OF 2 TIN | 4 | 2010-12-07 | 4.25 | 12347 | Iceland |
| 19 | 2010-12-07 | 537626 | 22729 | ALARM CLOC | 4 | 2010-12-07 | 3.75 | 12347 | Iceland |
| 27 | 2010-12-07 | 537626 | 22727 | ALARM CLOC | 4 | 2010-12-07 | 3.75 | 12347 | Iceland |
| 28 | 2010-12-07 | 537626 | 71477 | COLOUR GLA | 12 | 2010-12-07 | 3.25 | 12347 | Iceland |
| 29 | 2010-12-07 | 537626 | 22773 | GREEN DRAW | 12 | 2010-12-07 | 1.25 | 12347 | Iceland |
| 30 | 2010-12-07 | 537626 | 21171 | BATHROOM | 12 | 2010-12-07 | 1.45 | 12347 | Iceland |
| 37 | 2011-01-26 | 542237 | 21832 | CHOCOLATE | 12 | 2011-01-26 | 1.65 | 12347 | Iceland |
| 38 | 2011-01-26 | 542237 | 22417 | PACK OF 60 S | 24 | 2011-01-26 | 0.55 | 12347 | Iceland |
| 39 | 2011-01-26 | 542237 | 21154 | RED RETROS | 10 | 2011-01-26 | 1.25 | 12347 | Iceland |
| 40 | 2011-01-26 | 542237 | 22376 | AIRLINE BAG | 4 | 2011-01-26 | 4.25 | 12347 | Iceland |
| 46 | 2011-01-26 | 542237 | 22422 | TOOTHPAST | 12 | 2011-01-26 | 0.65 | 12347 | Iceland |
| 47 | 2011-01-26 | 542237 | 22729 | ALARM CLOC | 4 | 2011-01-26 | 3.75 | 12347 | Iceland |
| 48 | 2011-01-26 | 542237 | 84991 | 60 TEATIME F | 24 | 2011-01-26 | 0.55 | 12347 | Iceland |
| 49 | 2011-01-26 | 542237 | 22725 | ALARM CLOC | 4 | 2011-01-26 | 3.75 | 12347 | Iceland |
| 50 | 2011-01-26 | 542237 | 22196 | SMALL HEAR | 12 | 2011-01-26 | 0.85 | 12347 | Iceland |

• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT

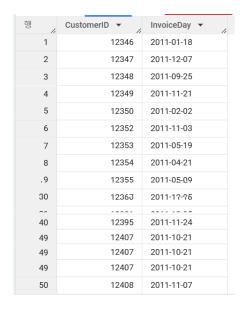
MAX(DATE(InvoiceDate)) OVER () AS most_recent_date, DATE(InvoiceDate) AS InvoiceDay,

FROM bold-syntax-456104-m4.modulabs_project.data;

| 행 // | most_recent_date | InvoiceDay | InvoiceNo | StockCode | Description | Quantity 🔻 | InvoiceDate | UnitPrice | CustomerID 🕶 | Country - |
|----------|--------------------------|--------------------|------------------|----------------|-------------|------------|---------------------|--------------|----------------|----------------------------------|
| 1 | 2011-12-09 | 2011-05 | 553546 | 23299 | FOOD CO | 50 | 2011-05 | 3.39 | 12415 | Australia |
| 2 | 2011-12-09 | 2011-06 | 556917 | 22750 | FELTCRA | 48 | 2011-06 | 3.39 | 12415 | Australia |
| 3 | 2011-12-09 | 2011-07 | 559919 | 23121 | PACK OF | 288 | 2011-07 | 0.36 | 12415 | Australia |
| 4 | 2011-12-09 | 2011-03 | 545226 | 47590A | BLUE HA | 3 | 2011-03 | 5.45 | 12428 | Finland |
| 5 | 2011-12-09 | 2011-12 | 581476 | 23429 | RED RET | 2 | 2011-12 | 8.15 | 12433 | Norway |
| 6 | 2011-12-09 | 2011-07 | 560991 | 21829 | DINOSA | 36 | 2011-07 | 0.21 | 12438 | Norway |
| 7 | 2011-12-09 | 2011-01 | 541518 | 18007 | ESSENTI | 24 | 2011-01 | 0.18 | 12451 | Switzerland |
| 8 | 2011-12-09 | 2011-02 | 544097 | 22855 | FINE WIC | 48 | 2011-02 | 1.06 | 12455 | Cyprus |
| 9 | 2011-12-09 | 2011-02 | 543117 | 22846 | BREAD B | 4 | 2011-02 | 14.95 | 12464 | Belgium |
| 17 | 2011-12-09 | 2011-06 | 555936 | 21084 | SET/6 C | 24 | 2011-06 | 0.19 | 12567 | France |
| 18 | 2011-12-09 | 2011-10 | 571670 | 23445 | ICE CRE | 200 | 2011-10 | 0.72 | 12611 | Italy |
| 19 | 2011-12-09 | 2011-08 | 563345 | 23293 | SET OF 1 | 128 | 2011-08 | 0.72 | 12619 | Germany |
| 20 | 2011-12-09 | 2011-10 | 569893 | 16045 | POPART | 200 | 2011-10 | 0.04 | 12627 | Germanv |
| 28 | 2011-12-09 | 2011-01 | 540351 | 21056 | DOCTOR' | 1 | 2011-01 | 8.95 | 12735 | France |
| 29 | 2011-12-09 | 2011-03 | 546366 | 21034 | REX CAS | 1 | 2011-03 | 0.95 | 12748 | United Kingdom |
| 30 | 2011-12-09 | 2011-07 | 558705 | 21386 | IVORY H | 3 | 2011-07 | 0.19 | 12748 | United Kingdom |
| 31 37 | 2011-12-09 2011-12-09 | 2011-11 2011-06 | 574470 556914 | 16237 79321 | CHILLI LI | 5 6 | 2011-11- 2011-06 | 0 21 5.75 | 12749 12877 | United Kingdom United Kingdom |
| 38 | 2011-12-09 | 2011-02 | 543829 | 22570 | FELTCRA | 192 | 2011-02 | 3.39 | 12939 | United Kingdom |
| 39 | 2011-12-09 | 2011-10 | 569858 | 21239 | PINK PO | 96 | 2011-10 | 0.72 | 12939 | United Kingdom |
| 40 | 2011-12-09 | 2011-10 | 571270 | 23163 | REGENC | 2 | 2011-10 | 2.49 | 12940 | United Kingdom |
| 46 | 2011-12-09 | 2011-03 | 547418 | 82484 | WOOD B | 2 | 2011-03 | 7.9 | 13148 | United Kingdom |
| 47 | 2011-12-09 | 2011-10 | 570240 | 46000M | POLYEST | 3 | 2011-10 | 1.55 | 13212 | United Kingdom |
| 48 | 2011-12-09 | 2011-09 | 568318 | 23119 | PACK OF | 12 | 2011-09 | 0.62 | 13233 | United Kingdom |
| 49 | 2011-12-09 | 2010-12 | 539981 | 85180A | RED HEA | 3 | 2010-12 | 4.65 | 13304 | United Kingdom |
| 46 | 2011-12-09 | 2011-03 | 547418 | 82484 | WOOD B | 2 | 2011-03 | 7.9 | 13148 | United Kingdom |
| 47 | 2011-12-09 | 2011-10 | 570240 | 46000M | POLYEST | 3 | 2011-10 | 1.55 | 13212 | United Kingdom |
| 48 | 2011-12-09 | 2011-09 | 568318 | 23119 | PACK OF | 12 | 2011-09 | 0.62 | 13233 | United Kingdom |
| 49 | 2011-12-09 | 2010-12 | 539981 | 85180A | RED HEA | 3 | 2010-12 | 4.65 | 13304 | United Kingdom |
| 50 | 2011-12-09 | 2011-07 | 561873 | 22750 | FELTCRA | 48 | 2011-07 | 3.39 | 13316 | United Kingdom |

• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM bold-syntax-456104-m4.modulabs_project.data
GROUP BY CustomerID;



• 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 bold-syntax-456104-m4.modulabs_project.data
GROUP BY CustomerID
);
```

| 행 // | CustomerID ▼ | recency ▼ |
|------|--------------|-----------|
| 1 | 12429 | 9 |
| 2 | 12512 | 66 |
| 3 | 12601 | 189 |
| 4 | 12644 | 129 |
| 5 | 13101 | 234 |
| 6 | 13390 | 74 |
| 7 | 13396 | 21 |
| 8 | 13500 | 23 |
| 9 | 13598 | 47 |
| 20 | 15306 | 64 |
| 31 | 16686 | 47 |
| 40 | 17811 | 4 |
| 49 | 13218 | 127 |
| 49 | 13218 | 127 |
| 50 | 14434 | 21 |

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

```
CREATE OR REPLACE TABLE bold-syntax-456104-m4.modulabs_project.user_r AS
WITH base AS (
SELECT
 CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM bold-syntax-456104-m4.modulabs_project.data
GROUP BY CustomerID
),
most_recent AS (
SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
FROM bold-syntax-456104-m4.modulabs_project.data
SELECT
b.CustomerID,
DATE_DIFF(m.most_recent_date, b.InvoiceDay, DAY) AS recency
FROM base b
CROSS JOIN most_recent m;
```

| 필드 이름 | 유형 | 모드 |
|------------|---------|----------|
| CustomerID | INTEGER | NULLABLE |
| recency | INTEGER | NULLABLE |

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM project_name.modulabs_project.data
GROUP BY CustomerID;

| 행 // | CustomerID ▼ | purchase_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 |
| 20 | 12367 | 1 |
| 21 | 12370 | 4 |

• 각 고객 별로 구매한 아이템의 총 수량 더하기

SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM bold-syntax-456104-m4.modulabs_project.data
GROUP BY CustomerID;

| 행 | CustomerID ▼ | item_cnt ▼ |
|----|--------------|------------|
| 11 | // | - // |
| 1 | 12346 | 0 |
| 2 | 12347 | 2458 |
| 3 | 12348 | 2332 |
| 4 | 12349 | 630 |
| 5 | 12350 | 196 |
| 6 | 12352 | 463 |
| 7 | 12353 | 20 |
| 8 | 12354 | 530 |
| 19 | 12365 | 173 |
| 30 | 12380 | 1109 |
| 39 | 12394 | 808 |
| 48 | 12406 | 1809 |
| 49 | 12407 | 1403 |
| 49 | 12407 | 1403 |
| 50 | 12408 | 1382 |

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

```
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
 CustomerID,
 COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM bold-syntax-456104-m4.modulabs_project.data
GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
 FROM bold-syntax-456104-m4.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 bold-syntax-456104-m4.modulabs_project.user_r AS ur
 ON pc.CustomerID = ur.CustomerID;
```

🖶 필터 속성 이름 또는 값 입력

| 필드 이름 | 유형 | 모드 | 7 |
|--------------|---------|----------|---|
| CustomerID | INTEGER | NULLABLE | - |
| purchase_cnt | INTEGER | NULLABLE | - |
| item_cnt | INTEGER | NULLABLE | - |
| recency | INTEGER | NULLABLE | - |

Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM bold-syntax-456104-m4.modulabs_project.data
GROUP BY CustomerID;
```

| 행 // | CustomerID ▼ | user_total ▼ |
|------|--------------|----------------------|
| 1 | 12346 | 0.0 |
| 2 | 12347 | 4310.0 |
| 3 | 12348 | 1437.2 |
| 4 | 12349 | 1457.6 |
| 5 | 12350 | 294.4 |
| 6 | 12352 | 1265.4 |
| 7 | 12353 | 89.0 |
| 8 | 12354 | 1079.4 |
| ۰,9 | 12355 | ₫ 59. |
| 30 | 1236ปี | ∠423.6 |
| | | |
| 40 | 12395 | 2662.3 |
| 49 | 12407 | 1507.1 |
| 49 | 12407 | 1507.1 |
| 49 | 12407 | 1507.1 |
| 50 | 12408 | 2587.6 |

• 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_name.modulabs_project.user_rfm AS
SELECT

rf.CustomerID AS CustomerID,
rf.purchase_cnt,
rf.item_cnt,
rf.recency,
ut.user_total,
# [[YOUR QUERY]] AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
-- 고객 별 총 지출액
SELECT
# [[YOUR QUERY]]
) ut
ON rf.CustomerID = ut.CustomerID;
```

| 〒 필터 속성 이름 또는 값 입력 | | | | |
|--------------------|--------------|---------|----------|--|
| | 필드 이름 | 유형 | 모드 | |
| | CustomerID | INTEGER | NULLABLE | |
| | purchase_cnt | INTEGER | NULLABLE | |
| | item_cnt | INTEGER | NULLABLE | |
| | recency | INTEGER | NULLABLE | |
| | user_total | FLOAT | NULLABLE | |
| | user_average | FLOAT | NULLABLE | |

RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

SELECT *
FROM bold-syntax-456104-m4.modulabs_project.user_rfm
ORDER BY CustomerID
LIMIT 100;

| 행 // | CustomerID ▼ | purchase_cnt 🕶 | item_cnt ▼ | recency ▼ | user_total ▼ | user_average ▼ |
|----------|----------------|----------------|------------|-----------|-----------------|----------------|
| 1 | 12346 | 2 | 0 | 325 | 0.0 | 0.0 |
| 2 | 12347 | 7 | 2458 | 2 | 4310.0 | 615.7 |
| 3 | 12348 | 4 | 2332 | 75 | 1437.2 | 359.3 |
| 4 | 12349 | 1 | 630 | 18 | 1457.6 | 1457.6 |
| 5 | 12350 | 1 | 196 | 310 | 294.4 | 294.4 |
| 6 | 12352 | 8 | 463 | 36 | 1265.4 | 158.2 |
| 7 | 12353 | 1 | 20 | 204 | 89.0 | 89.0 |
| 18 | 12364 | 4 | 1499 | 7 | 1208.1 | 302.0 |
| 29 38 | 12379 12393 | 3 4 | 401 816 | 81 72 | 775.3 1582.6 | 258.4 395.6 |
| 47 | 12405 | 1 | 849 | 148 | 1390.4 | 1390.4 |
| 49 | 12407 | 5 | 1403 | 49 | 1507.1 | 301.4 |
| 50 | 12408 | 8 | 1382 | 32 | 2587.6 | 323.4 |

11-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

• 1) 고객 별로 구매한 상품들의 고유한 수를 계산하기

2)

user_rfm 테이블과 결과를 합치기

3)

user_data 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS
WITH unique_products AS (
SELECT
CustomerID,
COUNT(DISTINCT StockCode) AS unique_products
FROM project_name.modulabs_project.data
GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 균 구매 소요 일수를 계산하고, 그 결과를 user_data 에 통합

CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS WITH purchase_intervals AS ($\,$

```
-- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
 SELECT
 CustomerID,
  CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
  -- (1) 구매와 구매 사이에 소요된 일수
  SELECT
   CustomerID,
  DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
  project_name.modulabs_project.data
  WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

3. 구매 취소 경향성

• 고객의 취소 패턴 파악하기

1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수

2) 취소 비율(cancel_rate): 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율

 취소 빈도와 취소 비율을 계산하고 그 결과를 user_data 에 통합하기 (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS

WITH TransactionInfo AS (
SELECT
CustomerID,
# [[YOUR QUERY]] AS total_transactions,
# [[YOUR QUERY]] AS cancel_frequency
FROM project_name.modulabs_project.data
# [[YOUR QUERY]]
)

SELECT u.*, t.* EXCEPT(CustomerID), # [[YOUR QUERY]] AS cancel_rate
FROM `project_name.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

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

```
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

회고

[회고 내용을 작성해주세요]

Keep : 그래도 복습을 많이 한 덕분에 대부분의 문제를 시간 내에 풀 수 있었다. Problem : 너무 간단한 오류를 가지고 너무 많이 헤멨던거 같다. 기초를 탄탄히 하자

Try : 실무적인 내용은 어느 정도 적응이 되는거 같으니 기초만 다시 조금 다지면 될 거 같다