고객을 세그먼테이션하자 [프로젝트] (1)

11-2. 데이터 불러오기

데이터 살펴보기

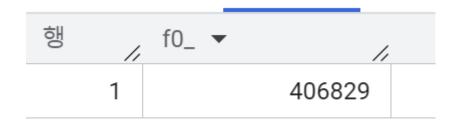
• 테이블에 있는 10개의 행만 출력하기

SELECT *
FROM dauntless-karma-466601-p1.modulabs_project.data
LIMIT 10



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT COUNT(*)
FROM dauntless-karma-466601-p1.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 -- 540455
, COUNT(Quantity) AS COUNT_Quantity
, COUNT(InvoiceDate) AS COUNT_InvoiceDate
, COUNT(UnitPrice) AS COUNT_UnitPrice
, COUNT(CustomerID) AS COUNT_CustomerID -- 406829
, COUNT(Country) AS COUNT_Country
FROM dauntless-karma-466601-p1.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) / COMMON COMMON

SELECT

'Description' AS column_name,

ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / (FROM dauntless-karma-466601-p1.modulabs_project.data UNION ALL

SELECT

'CustomerID' AS column_name,

ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / (FROM dauntless-karma-466601-p1.modulabs_project.data

135,080 행을 삭제한 후에 다시 캡쳐를 떠서 결과가 바뀌었습니다ㅠㅠ 양해바랍니다

작업 정보		결과	차트 .	JSON	실행 세부정보	실행 그래프		
행 //	행 column_name ▼ missing_percenta							
1	Description				0.0			
2	CustomerID				0.0			
3	InvoiceNo				0.0			

결측치 처리 전략

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

SELECT DISTINCT Description
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE StockCode = '85123A'

135,080 행을 삭제한 후에 다시 캡쳐를 떠서 결과가 줄었습니다



결측치 처리

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

DELETE FROM dauntless-karma-466601-p1.modulabs_project.data WHERE Description IS NULL OR CustomerID IS NULL

[결과 이미지를 넣어주세요]

● 이 문으로 data의 행 135,080개가 삭제되었습니다. 테이블로 이동

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- 중복된 행의 수를 세어보기
 - 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPri, COUNT(*) AS DUP

FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, Unit
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 dauntless-karma-466601-p1.modulabs_prc AS SELECT DISTINCT * FROM dauntless-karma-466601-p1.modulabs_pro

이 문으로 이름이 data인 테이블이 교체되었습니다.

테이블로 이동

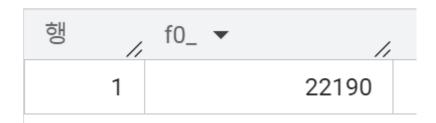
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT COUNT(DISTINCT InvoiceNo)

FROM dauntless-karma-466601-p1.modulabs_project.data



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo FROM dauntless-karma-466601-p1.modulabs_project.data LIMIT 100



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

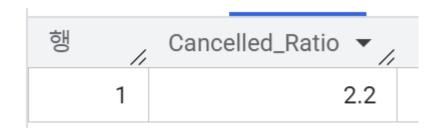
SELECT *
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100



• 구매 건 상태가 Canceled 인 데이터의 비율(%) - 소수점 첫번째 자리까지

SELECT

ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / C FROM dauntless-karma-466601-p1.modulabs_project.data

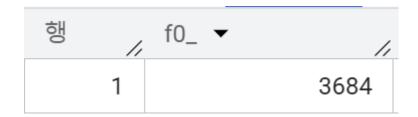


StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

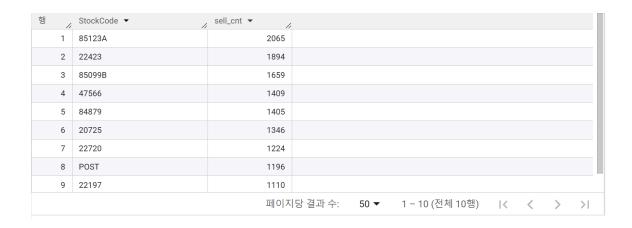
SELECT COUNT(DISTINCT StockCode)
FROM dauntless-karma-466601-p1.modulabs_project.data

[결과 이미지를 넣어주세요]



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

SELECT StockCode, COUNT(*) AS sell_cnt
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10



- 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]', FROM dauntless-karma-466601-p1.modulabs_project.data
)
WHERE number_count BETWEEN 0 AND 1
```

[결과 이미지를 넣어주세요]

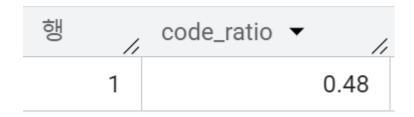
행 //	StockCode ▼	number_count ▼ //
1	POST	0
2	М	0
3	C2	1
4	D	0
5	BANK CHARGES	0
6	PADS	0
7	DOT	0
8	CRUK	0

- 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(SUM(CASE WHEN StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CI / COUNT(StockCode) * 100, 2) AS code_ratio

FROM dauntless-karma-466601-p1.modulabs_project.data



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE StockCode IN (
SELECT DISTINCT StockCode
FROM (
SELECT DISTINCT StockCode, number_count
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[r])
FROM dauntless-karma-466601-p1.modulabs_project.data
)
WHERE number_count BETWEEN 0 AND 1
)
)
```

[결과 이미지를 넣어주세요]

① 민으로 data의 행 1,915개가 삭제되었습니다. 데이블로 이동

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT Description, COUNT(*) AS description_cnt FROM dauntless-karma-466601-p1.modulabs_project.data GROUP BY Description ORDER BY description_cnt DESC LIMIT 30



• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE

FROM dauntless-karma-466601-p1.modulabs_project.data WHERE Description IN('Next Day Carriage', 'Next Day Carriage')

[결과 이미지를 넣어주세요]

① 민으로 data의 행 80개가 삭제되었습니다. 테이블로 이동

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE dauntless-karma-466601-p1.modulabs_proje SELECT

* EXCEPT (Description),

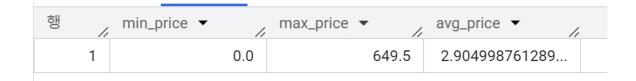
UPPER(Description) AS Description FROM dauntless-karma-466601-p1.modulabs_project.data

① 이 문으로 이름이 data인 테이블이 교체되었습니다. 테이블로 이동

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 dauntless-karma-466601-p1.modulabs_project.data

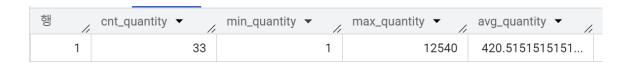


• 단가가 O원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기

SELECT COUNT(Quantity) AS cnt_quantity

- , MIN(Quantity) AS min_quantity
- , MAX(Quantity) AS max_quantity
- , AVG(Quantity) AS avg_quantity

FROM dauntless-karma-466601-p1.modulabs_project.data WHERE UnitPrice = 0



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE dauntless-karma-466601-p1.modulabs_proje SELECT *

FROM dauntless-karma-466601-p1.modulabs_project.data WHERE UnitPrice <> 0

① 마 문으로 이름이 data인 테이블이 교체되었습니다.

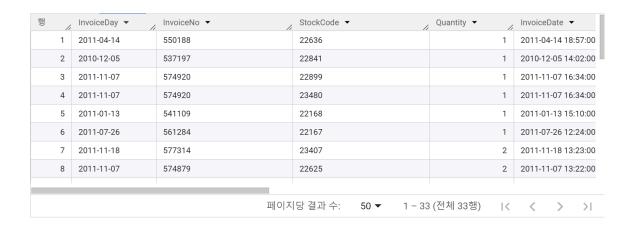
테이블로 이동

11-7. RFM 스코어

Recency

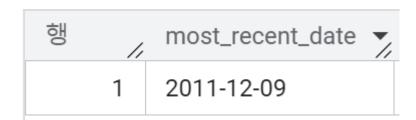
• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM dauntless-karma-466601-p1.modulabs_project.data



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT DATE(MAX(InvoiceDate)) AS most_recent_date FROM dauntless-karma-466601-p1.modulabs_project.data



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

• 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산 하기

```
SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recenc
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM project_name.modulabs_project.data
GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE dauntless-karma-466601-p1.modulabs_proje SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recenc FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY CustomerID
);
```

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
```

COUNT(DISTINCT InvoiceNo) AS purchase_cnt FROM dauntless-karma-466601-p1.modulabs_project.data GROUP BY CustomerID

[결과 이미지를 넣어주세요]

행 //	CustomerID ▼	purchase_cnt ▼ //		
1	12457	1		
2	12647	1		
3	13985	1		
4	15107	1		
5	16818	1		
6	12444	1		
7	13014	1		
8	14410	1		
9	15804	1		
			페이지당 결과 수: 50 ▼ 1 - 24 (전체 24행) < < > >	

• 각 고객 별로 구매한 아이템의 총 수량 더하기

SELECT

CustomerID,

SUM(Quantity) AS item_cnt

FROM dauntless-karma-466601-p1.modulabs_project.data GROUP BY CustomerID



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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
# [[YOUR QUERY]]
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
 # [[YOUR QUERY]]
)
-- 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_name.modulabs_project.user_r AS ur
 ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

● 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

테이블로 이동

Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

SELECT CustomerID, ROUND(SUM(UnitPrice * Quantity), 1) AS user_total FROM dauntless-karma-466601-p1.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							
9	12355	459.4							
			페이지당 결과 수:	50 ▼	1 - 50 (전체 4362행)	<	<	>	>1

• 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 dauntless-karma-466601-p1.modulabs_proje
SELECT
rf.CustomerID AS CustomerID,
rf.purchase_cnt,
rf.item_cnt,
rf.recency,
ut.user_total,
ut.user_total / rf.purchase_cnt AS user_average
FROM dauntless-karma-466601-p1.modulabs_project.user_rf rf
LEFT JOIN (
-- 고객 별 총 지출액
SELECT
CustomerID,
ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
```

FROM dauntless-karma-466601-p1.modulabs_project.data GROUP BY CustomerID

) ut

ON rf.CustomerID = ut.CustomerID ORDER BY rf.purchase_cnt DESC

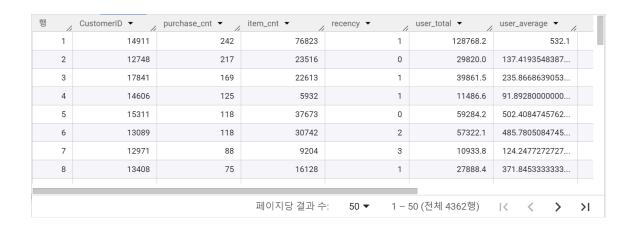
● 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

테이블로 이동

RFM 통합 테이블 출력하기

최종 user_rfm 테이블을 출력하기

SELECT * FROM dauntless-karma-466601-p1.modulabs_project.user_rfm



11-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

1) 고객 별로 구매한 상품들의 고유한 수를 계산하기2)

```
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;
```

● 이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

테이블로 이동

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 FROM (
-- (1) 구매와 구매 사이에 소요된 일수
SELECT
```

```
CustomerID,
DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY Custome FROM
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;
```

이 문으로 이름이 user_data인 테이블이 교체되었습니다.

테이블로 이동

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 dauntless-karma-466601-p1.modulabs_project.u
WITH TransactionInfo AS (
    SELECT CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_free
FROM dauntless-karma-466601-p1.modulabs_project.data
    GROUP BY CustomerID
)
```

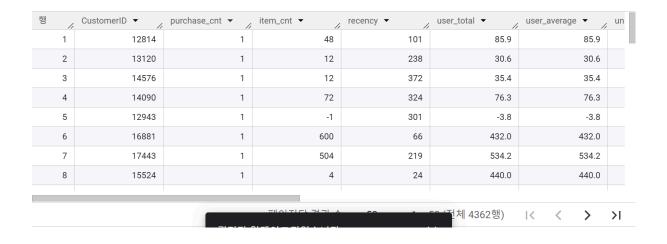
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_trar FROM `dauntless-karma-466601-p1.modulabs_project.user_data` AS u LEFT JOIN TransactionInfo AS t ON u.CustomerID = t.CustomerID

[결과 이미지를 넣어주세요]

① 민으로 이름이 user_data인 테이블이 교체되었습니다. 테이블로 이동

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

SELECT * FROM dauntless-karma-466601-p1.modulabs_project.user_data



회고

[회고 내용을 작성해주세요]

Keep: 하루동안 특정 지점에서 오래 멈추지않고 마무리로 진전을 이루었다는것

Problem : 문제풀이, 코드복사, 결과화면 복사에 집중하다보니 정작 중요한 결과물의 검증을 매 단계별로 꼼꼼히 하지 못했다. 그래서 데이터를 3번이나 다시 밀어넣으며 노가다로 검증을 했다.(지금도 결과가 100% 맞게 되었는지는 다 확인하지 못한 상황)

Try: 좀더 여유를 갖고 처음부터 잘 검증한다