# 고객을 세그먼테이션하자! [프로젝트] - 김건우

# 11-2. 데이터 불러오기

#### 데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM hip-light-470203-q0.modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)
FROM hip-light-470203-q0.modulabs_project.data
;
[결과 이미지를 넣어주세요]
```



#### 데이터 수 세기

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

```
SELECT COUNT(InvoiceNo), COUNT(StockCode), COUNT(Description), COUNT(Quantity), COUNT(InvoiceDate), COUNT(UnitPrice), COUNT(CustomerID), COUNT(Country) FROM hip-light-470203-q0.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) AS missing\_percentage FROM hip-light-470203-q0.modulabs\_project.data UNION ALL

#### **SELECT**

'StockCode' AS column\_name,

ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS missing\_percentage FROM hip-light-470203-q0.modulabs\_project.data UNION ALL

#### SELECT

'Description' AS column\_name,

 $ROUND (SUM (CASE \ WHEN \ Description \ IS \ NULL \ THEN \ 1 \ ELSE \ 0 \ END) \ / \ COUNT (*) * 100, \ 2) \ AS \ missing\_percentage \\ FROM \ hip-light-470203-q0.modulabs\_project.data \ UNION \ ALL$ 

#### **SELECT**

'Quantity' AS column\_name,

ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS missing\_percentage FROM hip-light-470203-q0.modulabs\_project.data UNION ALL

#### **SELECT**

'InvoiceDate' AS column\_name,

ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS missing\_percentage FROM hip-light-470203-q0.modulabs\_project.data UNION ALL

#### **SELECT**

'UnitPrice' AS column\_name,

 $ROUND (SUM (CASE \ WHEN \ UnitPrice \ IS \ NULL \ THEN \ 1 \ ELSE \ 0 \ END) \ / \ COUNT (*) * 100, \ 2) \ AS \ missing\_percentage \\ FROM \ hip-light-470203-q0.modulabs\_project.data \ UNION \ ALL$ 

#### **SELECT**

'CustomerID' AS column\_name,

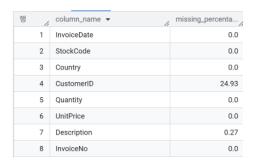
ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS missing\_percentage FROM hip-light-470203-q0.modulabs\_project.data UNION ALL

#### SELECT

'Country' AS column\_name,

 $ROUND(SUM(CASE\ WHEN\ Country\ IS\ NULL\ THEN\ 1\ ELSE\ 0\ END)\ /\ COUNT(*)\ *\ 100,\ 2)\ AS\ missing\_percentage$  FROM hip-light-470203-q0.modulabs\\_project.data ;

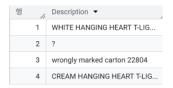
#### [결과 이미지를 넣어주세요]



#### 결측치 처리 전략

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

```
SELECT Description
FROM hip-light-470203-q0.modulabs_project.data
WHERE StockCode = '85123A'
GROUP BY 1
;
```



## 결측치 처리

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

```
DELETE FROM hip-light-470203-q0.modulabs_project.data
WHERE CustomerID is NULL
OR Description is NULL
;
```

[결과 이미지를 넣어주세요]

- ① 마문으로 data의 행 133,626개가 삭제되었습니다.
- 이 문으로 data의 행 1,454개가 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

```
WITH bok AS (
SELECT COUNT(*)AS a
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING a > 1)
SELECT COUNT(a)
FROM bok
;
```

행 , f0\_ ▼ /, 1 4837

## 중복값 처리

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

```
-- 중복값 처리
CREATE OR REPLACE TABLE hip-light-470203-q0.modulabs_project.data AS -- 테이블 삭제후 다시 생성
SELECT DISTINCT *
FROM hip-light-470203-q0.modulabs_project.data
;
-- 중복값 처리 후 남은 행 개수
SELECT COUNT(*)
FROM hip-light-470203-q0.modulabs_project.data
;
```



# 11-6. 데이터 전처리(3): 오류값 처리

#### InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT .InvoiceNo)AS b,
FROM hip-light-470203-q0.modulabs_project.data
;
```

[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM hip-light-470203-q0.modulabs_project.data
LIMIT 100
;
```



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

SELECT \*
FROM hip-light-470203-q0.modulabs\_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;

[결과 이미지를 넣어주세요]



• 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) FROM hip-light-470203-q0.modulabs\_project.data;

[결과 이미지를 넣어주세요]



## StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

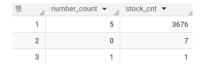
```
SELECT COUNT(DISTINCT StockCode)AS c,
FROM hip-light-470203-q0.modulabs_project.data
;
```

[결과 이미지를 넣어주세요]



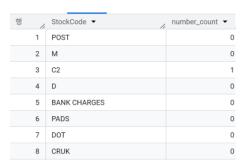
- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기
  - 。 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
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]', '')) AS number_count
FROM hip-light-470203-q0.modulabs_project.data
)
WHERE number_count <2
;
```



- 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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
SELECT DISTINCT StockCode, number_count
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM project_name.modulabs_project.data
)
WHERE # [[YOUR QUERY]];

--아래가 내가 푼 방법
SELECT ROUND(SUM(CASE WHEN number_count <2 THEN 1 ELSE 0 END)/ COUNT(*)*100, 2)
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM hip-light-470203-q0.modulabs_project.data
)
;
```

[결과 이미지를 넣어주세요]



• 제품과 관련되지 않은 거래 기록을 제거하기

DELETE FROM hip-light-470203-q0.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 hip-light-470203-q0.modulabs_project.data
)

WHERE number_count <2
);
```

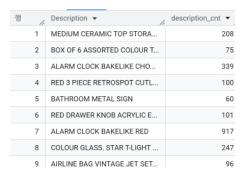
이 문으로 data의 행 1,915개가 삭제되었습니다.

## Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
LIMIT 30
;
```

[결과 이미지를 넣어주세요]



• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM hip-light-470203-q0.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

[결과 이미지를 넣어주세요]

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE hip-light-470203-q0.modulabs_project.data AS
SELECT

* EXCEPT (Description),
UPPER(Description) AS Description
FROM hip-light-470203-q0.modulabs_project.data
;
```



## UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

SELECT MIN(UnitPrice) AS min\_price, MAX(UnitPrice) AS max\_price, AVG(UnitPrice) AS avg\_price FROM hip-light-470203-q0.modulabs\_project.data;

[결과 이미지를 넣어주세요]



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

SELECT COUNT(\*) AS cnt\_quantity, MIN(Quantity) AS min\_quantity, MAX(Quantity) AS max\_quantity, AVG(Quantity) AS avg\_quantity
FROM hip-light-470203-q0.modulabs\_project.data
WHERE UnitPrice=0
;

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE hip-light-470203-q0.modulabs_project.data AS SELECT *
FROM hip-light-470203-q0.modulabs_project.data
WHERE UnitPrice=0
;
```

[결과 이미지를 넣어주세요]

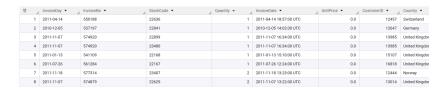
● 이 문으로 이름이 data인 테이블이 교체되었습니다.

# 11-7. RFM 스코어

#### Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE (InvoiceDate) AS InvoiceDay, *
FROM hip-light-470203-q0.modulabs_project.data
;
```



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
MAX(InvoiceDate) AS most_recent_date,
DATE(MAX(InvoiceDate)) AS InvoiceDay,
FROM hip-light-470203-q0.modulabs_project.data
;
```

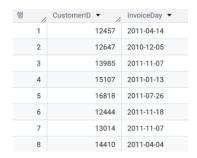
 ∰
 most\_recent\_date ▼
 // InvoiceDay ▼

 1
 2011-11-25 15:57:00 UTC
 2011-11-25

• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
;
```

[결과 이미지를 넣어주세요]



• 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
);
```

• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user\_r 이라는 이름의 테이블로 저장하기

```
CREATE OR REPLACE TABLE hip-light-470203-q0.modulabs_project.user_r AS
SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user\_r인 새 테이블이 생성되었습니다.

## **Frequency**

• 고객마다 고유한 InvoiceNo의 수를 세어보기

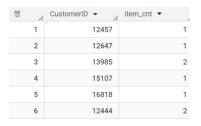
```
SELECT
CustomerID,
COUNT(InvoiceNo) AS purchase_cnt
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
;
```

[결과 이미지를 넣어주세요]

행 //	CustomerID	• //	purchase_cnt ▼	,
1		12457		1
2		12647		1
3		13985		2
4		15107		1
5		16818		1
6		12444		1
7		13014		1

• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
;
```



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

```
CREATE OR REPLACE TABLE hip-light-470203-q0.modulabs_project.user_rf AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
CustomerID,
COUNT(InvoiceNo) AS purchase_cnt
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
)
-- 기존의 user_r에 (1)과 (2)를 통합 저장
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 hip-light-470203-q0.modulabs_project.user_r AS ur
ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

● 이 문으로 이름이 user\_rf인 새 테이블이 생성되었습니다.

#### **Monetary**

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(UnitPrice * Quantity),1)AS user_total
FROM hip-light-470203-q0.modulabs_project.data
```

```
GROUP BY 1;
```

행 //	CustomerID	· //	user_total ▼
1		12346	0.0
2		12347	4310.0
3		12348	1437.2
4		12349	1457.6

- 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 hip-light-470203-q0.modulabs_project.user_rfm AS
SELECT
rf.CustomerID AS CustomerID,
rf.purchase_cnt,
rf.item_cnt,
rf.recency,
ut.user_total,
ROUND(ut.user_total / rf.purchase_cnt, 1) AS user_average
FROM hip-light-470203-q0.modulabs_project.user_rf rf
LEFT JOIN (
-- 고객 별 총 지출액
SELECT
CustomerID,
ROUND(SUM(UnitPrice * Quantity),1)AS user_total
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY 1
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user\_rfm인 새 테이블이 생성되었습니다.

## RFM 통합 테이블 출력하기

• 최종 user\_rfm 테이블을 출력하기

SELECT \* FROM hip-light-470203-q0.modulabs\_project.user\_rfm;



# 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 hip-light-470203-q0.modulabs_project.user_data AS
WITH unique_products AS (
SELECT
CustomerID,
COUNT(DISTINCT StockCode) AS unique_products
FROM hip-light-470203-q0.modulabs_project.data
GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM hip-light-470203-q0.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

● 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 hip-light-470203-q0.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_
   hip-light-470203-q0.modulabs_project.data
  WHERE CustomerID IS NOT NULL
 GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM hip-light-470203-q0.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

• 고객의 취소 패턴 파악하기

3. 구매 취소 경향성

- 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 hip-light-470203-q0.modulabs_project.user_data AS
WITH TransactionInfo AS (
 SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS total_transactions,
  COUNT(DISTINCT IF(STARTS_WITH(InvoiceNo, 'C'), InvoiceNo, NULL)) AS cancel_frequency
 FROM hip-light-470203-q0.modulabs_project.data
 GROUP BY CustomerID
)
SELECT
 u.*,
 t.* EXCEPT(CustomerID),
 ROUND(IFNULL(t.cancel_frequency, 0) / NULLIF(t.total_transactions, 0), 2) AS cancel_rate
FROM hip-light-470203-q0.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

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 hip-light-470203-q0.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

8 /	CustomerID ▼	purchase_ont ▼	item_cnt ▼	recency •	user_total ▼	user_average ▼	unique_products •	average_interval • //	total_transactions >	cancel_frequency 🔨	canceLrate ▼
5	14340	6	58	218	134.7	22.4	6	0.0	1	0	0.0
6	17879	6	45	173	178.5	29.8	6	0.0	1	0	0.0
7	15432	10	65	23	171.2	17.1	10	0.0	1	0	0.0
8	16096	12	155	264	320.6	26.7	12	0.0	1	0	0.0
9	14484	14	210	50	321.9	23.0	14	0.0	1	0	0.0
10	12436	14	203	99	416.4	29.7	14	0.0	1	0	0.0
	*****		***	~	200 7	20.0					

# 회고

[회고 내용을 작성해주세요]

Keep:

Problem:

Try: