# 고객을 세그먼테이션하자 [프로젝트] (1)

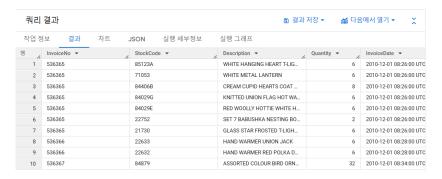
## 11-2. 데이터 불러오기

#### 데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

select \*
from infinite-chain-456201-u5.modulabs\_project.data
limit 10

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

select count(\*)
from infinite-chain-456201-u5.modulabs\_project.data

[결과 이미지를 넣어주세요]

#### 쿼리 결과



#### 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 infinite-chain-456201-u5.modulabs\_project.data



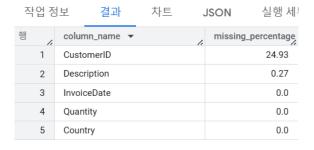
## 11-4. 데이터 전처리 방법(1): 결측치 제거

#### 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 *
FROM(
(SELECT 'InvoiceNo'AS column_name,
   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM infinite-chain-456201-u5.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 infinite-chain-456201-u5.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 infinite-chain-456201-u5.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 infinite-chain-456201-u5.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 infinite-chain-456201-u5.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 infinite-chain-456201-u5.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 infinite-chain-456201-u5.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 infinite-chain-456201-u5.modulabs_project.data)
ORDER BY 2 DESC
```

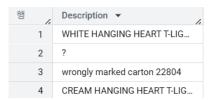
[결과 이미지를 넣어주세요]



#### 결측치 처리 전략

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

SELECT DISTINCT Description
FROM infinite-chain-456201-u5.modulabs\_project.data
WHERE StockCode = '85123A'



#### 결측치 처리

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

DELETE
FROM infinite-chain-456201-u5.modulabs\_project.data
WHERE CustomerID IS NULL
OR Description IS NULL

[결과 이미지를 넣어주세요]



## 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 \*
FROM infinite-chain-456201-u5.modulabs\_project.data
GROUP BY InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
HAVING COUNT(\*) > 1

[결과 이미지를 넣어주세요]

∘ 실수로 중값값 처리 이전 단계의 캡처본(중복 4837행)을 날려버렸고, 중복값 제거 이후에 실행 시 다음처럼 아무 것도 안 뜹니다. (중복 데이터 없음)



#### 중복값 처리

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

CREATE OR REPLACE TABLE infinite-chain-456201-u5.modulabs\_project.data AS SELECT DISTINCT \* FROM infinite-chain-456201-u5.modulabs\_project.data



작업 정보 <u>결과</u> 실행 세부정보 실행 그래프

● 이 문으로 이름이 data인 테이블이 교체되었습니다.

## 11-6. 데이터 전처리(3): 오류값 처리

#### InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT COUNT(distinct InvoiceNo)
FROM infinite-chain-456201-u5.modulabs\_project.data

[결과 이미지를 넣어주세요]



22190

1

• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT distinct InvoiceNo FROM infinite-chain-456201-u5.modulabs\_project.data limit 100

[결과 이미지를 넣어주세요]



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

SELECT \*
FROM infinite-chain-456201-u5.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) FROM infinite-chain-456201-u5.modulabs\_project.data

[결과 이미지를 넣어주세요]

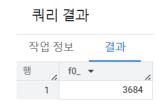


#### StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode)
FROM infinite-chain-456201-u5.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 infinite-chain-456201-u5.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]', '')) AS number_count
FROM infinite-chain-456201-u5.modulabs_project.data
)
WHERE number_count < 5;
```

쿼리	결과
----	----

작업 정	성보 결과	차트 .	JSON 실현	행 세-
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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

SELECT ROUND(SUM(CASE WHEN StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK') THEN 1 ELSE FROM infinite-chain-456201-u5.modulabs\_project.data

[결과 이미지를 넣어주세요]



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM infinite-chain-456201-u5.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'[0-9]', '')) AS number_count

FROM infinite-chain-456201-u5.modulabs_project.data
)

WHERE number_count < 5
```

#### Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM infinite-chain-456201-u5.modulabs_project.data
GROUP BY Description
ORDER BY 2 DESC
LIMIT 30;
```

이 문으로 data의 행 1,915개가 삭제되었습니다.

[결과 이미지를 넣어주세요]



• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE Description IN (
SELECT DISTINCT Description
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]')
AND Description IN ("High Resolution Image","Next Day Carriage")
)
```

[결과 이미지를 넣어주세요]



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE infinite-chain-456201-u5.modulabs\_project.data AS SELECT

\* EXCEPT (Description),
UPPER(Description) AS Description
FROM infinite-chain-456201-u5.modulabs\_project.data;

[결과 이미지를 넣어주세요]



#### 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 infinite-chain-456201-u5.modulabs\_project.data;

[결과 이미지를 넣어주세요]



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

```
SELECT count(Quantity) AS cnt_quantity,
min(Quantity) AS min_quantity,
max(Quantity) AS max_quantity,
avg(Quantity) AS avg_quantity
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE infinite-chain-456201-u5.modulabs_project.data AS (
SELECT *
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE UnitPrice <> 0
)
```



작업 정보 결과 실행 세부정보 실행 그래프 이 문으로 이름이 data인 테이블이 교체되었습니다.

#### 11-7. RFM 스코어

#### Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay, \* FROM infinite-chain-456201-u5.modulabs\_project.data

[결과 이미지를 넣어주세요]



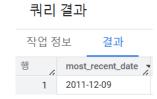
• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT

MAX(DATE(InvoiceDate)) AS most\_recent\_date

FROM infinite-chain-456201-u5.modulabs\_project.data

[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT
CustomerID,
DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM infinite-chain-456201-u5.modulabs\_project.data
GROUP BY CustomerID

#### 쿼리 결과

작업 정	<b>보</b> 결과	차트 JSON
행 //	CustomerID ▼	InvoiceDay ▼
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
	10050	0011 11 00

• 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 infinite-chain-456201-u5.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



#### **Frequency**

• 고객마다 고유한 InvoiceNo의 수를 세어보기

SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase\_cnt
FROM infinite-chain-456201-u5.modulabs\_project.data
GROUP BY 1

[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

SELECT CustomerID,
SUM(Quantity) AS item\_cnt
FROM infinite-chain-456201-u5.modulabs\_project.data
GROUP BY 1

[결과 이미지를 넣어주세요]



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

CREATE OR REPLACE TABLE infinite-chain-456201-u5.modulabs\_project.user\_rf AS (
-- (1) 전체 거래 건수 계산
WITH purchase\_cnt AS (
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase\_cnt
FROM infinite-chain-456201-u5.modulabs\_project.data
GROUP BY 1

```
),
 -- (2) 구매한 아이템 총 수량 계산
 item_cnt AS (
  SELECT CustomerID,
    SUM(Quantity) AS item_cnt
  FROM infinite-chain-456201-u5.modulabs_project.data
  GROUP BY 1
 -- 기존의 user_r에 (1)과 (2)를 통합
  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 infinite-chain-456201-u5.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID
)
```



#### **Monetary**

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(sum(UnitPrice * Quantity),1) AS user_total
FROM infinite-chain-456201-u5.modulabs_project.data
GROUP BY 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 infinite-chain-456201-u5.modulabs_project.user_rfm AS (
 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 infinite-chain-456201-u5.modulabs_project.user_rf rf
 LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
  CustomerID,
   ROUND(sum(UnitPrice * Quantity),1) AS user_total
  FROM infinite-chain-456201-u5.modulabs_project.data
  GROUP BY 1
) ut
ON rf.CustomerID = ut.CustomerID
```

# 쿼리 결과 작업 정보 결과 실행 세부정보 실행 그래프 ● 이 문으로 이름이 user\_rfm인 새 테이블이 생성되었습니다.

#### RFM 통합 테이블 출력하기

• 최종 user\_rfm 테이블을 출력하기

```
SELECT *
FROM infinite-chain-456201-u5.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]



## 11-8. 추가 Feature 추출

## 1. 구매하는 제품의 다양성

1) 고객 별로 구매한 상품들의 고유한 수를 계산하기
2)
 user\_rfm 테이블과 결과를 합치기

user\_data 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE infinite-chain-456201-u5.modulabs_project.user_data AS (
WITH unique_products AS (
SELECT
CustomerID,
COUNT(DISTINCT StockCode) AS unique_products
FROM infinite-chain-456201-u5.modulabs_project.data
GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM infinite-chain-456201-u5.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID
)
```



#### 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 infinite-chain-456201-u5.modulabs_project.user_data AS (
  WITH purchase_intervals AS (
   -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
   SELECT
    CustomerID,
    CASE
      WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0
      ELSE ROUND(AVG(interval_), 2)
    END AS average_interval
   FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
    infinite-chain-456201-u5.modulabs_project.data
    WHERE CustomerID IS NOT NULL
   GROUP BY CustomerID
  SELECT u.*, pi.* EXCEPT (CustomerID)
  FROM infinite-chain-456201-u5.modulabs_project.user_data AS u
  LEFT JOIN purchase_intervals AS pi
  ON u.CustomerID = pi.CustomerID
```

# 쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

● 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 project_name.modulabs_project.user_data AS (

WITH TransactionInfo AS (

SELECT

CustomerID,

count(InvoiceNo) AS total_transactions,

SUM(CASE WHEN InvoiceNo like "C%" THEN 1 ELSE 0 END) AS cancel_frequency

FROM infinite-chain-456201-u5.modulabs_project.data

GROUP BY 1
)

SELECT u.*, t.* EXCEPT(CustomerID), round(t.cancel_frequency / t.total_transactions * 100, 2) AS cancel_rate

FROM infinite-chain-456201-u5.modulabs_project.user_data AS u
```

[결과 이미지를 넣어주세요]

LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID

 쿼리 결과

 작업 정보
 결과
 실행 세부정보
 실행 그래프

 ●
 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 infinite-chain-456201-u5.modulabs\_project.user\_data



## 회고

[회고 내용을 작성해주세요]

Keep : 간신히 완성했다.

Problem : 해석에 대해 깊게 고려해볼 시간이 부족하다.

Try : 결과물에 대한 단계별 회고가 필요하다.