SQL_11. 고객을 세그먼테이션하자![프로젝트]

```
-- 데이터 살펴보기
SELECT *
FROM `modulabs_project.data`
LIMIT 10

-- 데이터 행 구성 보기
SELECT COUNT(*) AS row_count
FROM `modulabs-project-465302.modulabs_project.data`

-- 데이터 수 세기
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 `modulabs-project-465302.modulabs_project.data`

-- 데이터 전처리 _ 결측치 제거
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*
FROM `modulabs-project-465302.modulabs_project.data`

-- 결측치 알아보기
-- 다른 컬럼에도 동일하게 반영한 후, UNION ALL로 연결
SELECT
```

```
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*)
FROM `modulabs-project-465302.modulabs_project.data`

UNION ALL

SELECT
  'StockCode',
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*
FROM `modulabs-project-465302.modulabs_project.data`

UNION ALL

SELECT
  'Description',
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(
FROM `modulabs-project-465302.modulabs_project.data`

UNION ALL

SELECT
  'Quantity',
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) *
FROM `modulabs-project-465302.modulabs_project.data`

UNION ALL

SELECT
  'InvoiceDate',
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(
FROM `modulabs-project-465302.modulabs_project.data`

UNION ALL

SELECT
  'UnitPrice',
```

```
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*)
FROM `modulabs-project-465302.modulabs_project.data`

UNION ALL

SELECT
  'CustomerID',
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(
FROM `modulabs-project-465302.modulabs_project.data`

UNION ALL

SELECT
  'Country',
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) *
FROM `modulabs-project-465302.modulabs_project.data`

-- 참고!결측치 비율을 계산하는 방법은 위에서 제시한 CASE WHEN을 사용하는 것 외에도
SELECT column_name,
       ROUND((total - column_value) / total * 100, 2) AS missing_percentage
FROM (
  SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, CO
  SELECT 'StockCode', COUNT(StockCode), COUNT(*) FROM `modulabs-project
  SELECT 'Description', COUNT(Description), COUNT(*) FROM `modulabs-projec
  SELECT 'Quantity', COUNT(Quantity), COUNT(*) FROM `modulabs-project-465
  SELECT 'InvoiceDate', COUNT(InvoiceDate), COUNT(*) FROM `modulabs-proje
  SELECT 'UnitPrice', COUNT(UnitPrice), COUNT(*) FROM `modulabs-project-46
  SELECT 'CustomerID', COUNT(CustomerID), COUNT(*) FROM `modulabs-proje
  SELECT 'Country', COUNT(Country), COUNT(*) FROM `modulabs-project-4653
) AS column_data

-- 같은 제품(StockCode)이 항상 같은 상세 설명(Description)을 가지고 있지 않다는 데이
-- StockCode = '85123A'의 Description을 추출하는 쿼리문을 작성
SELECT Description
FROM `modulabs-project-465302.modulabs_project.data`
WHERE StockCode = '85123A'
```

```
-- 결측치 처리
DELETE FROM `modulabs-project-465302.modulabs_project.data`
WHERE InvoiceNo IS NULL
   OR StockCode IS NULL
   OR Description IS NULL
   OR Quantity IS NULL
   OR InvoiceDate IS NULL
   OR UnitPrice IS NULL
   OR CustomerID IS NULL
   OR Country IS NULL
-- data의 행 135,080개가 삭제됨


-- 데이터 전처리_ 중복값 확인
-- 중복된 행의 수를 세기
SELECT
  COUNT(*) AS duplicate_count
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS cnt
  FROM `modulabs-project-465302.modulabs_project.data`
  GROUP BY
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, Custome
  HAVING COUNT(*) > 1
)

-- 중복값 처리
```

```
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.data`
SELECT DISTINCT *
FROM `modulabs-project-465302.modulabs_project.data`

-- 중복값 처리 이후 남은 행의 개수
SELECT COUNT(*) AS row_count
FROM `modulabs-project-465302.modulabs_project.data`

-- 데이터 전처리_오류값 처리
-- InvoiceNo 살펴보기

-- 고유(unique)한 InvoiceNo의 개수를 출력
SELECT
  COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `modulabs-project-465302.modulabs_project.data`

-- 고유한 InvoiceNo를 100개를 출력
SELECT DISTINCT InvoiceNo
FROM `modulabs-project-465302.modulabs_project.data`
LIMIT 100

-- InvoiceNo가 'C'로 시작하는 행을 필터링
SELECT *
FROM `modulabs-project-465302.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;

-- 구매 건 상태가 Canceled 인 데이터의 비율(%)
SELECT ROUND(
  SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)
  / COUNT(*) * 100,
  1
) AS canceled_rate_percentage
FROM `modulabs-project-465302.modulabs_project.data`

-- StockCode 살펴보기
```

```
-- 고유한 StockCode의 개수를 출력
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `modulabs-project-465302.modulabs_project.data`

-- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력(상위 1
SELECT
  StockCode,
  COUNT(*) AS frequency
FROM `modulabs-project-465302.modulabs_project.data`
GROUP BY StockCode
ORDER BY frequency DESC
LIMIT 10

--이상치들이 몇 개나 있는지 확인하기 위하여 StockCode의 문자열 내 숫자의 길이 출력
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM `modulabs-project-465302.modulabs_project.data`
)
SELECT
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS
  COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC
-- 출력 결과를 보면, 8개를 제외하곤 StockCode에 5개의 숫자들이 포함되어 있는 것을 알
-- 숫자가 0개인 코드는 7개, 숫자가 1개인 코드는 1개

-- 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지를 확인
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS
  FROM `modulabs-project-465302.modulabs_project.data`
)
```

```
WHERE number_count <= 1

-- 데이터 수는 전체 데이터 수 대비 몇 퍼센트?
SELECT
  ROUND(
    COUNTIF(
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <
    ) / COUNT(*) * 100,
    2
  ) AS special_stockcode_percentage
FROM `modulabs-project-465302.modulabs_project.data`

-- 제품과 관련되지 않은 거래 기록을 제거
DELETE FROM `modulabs-project-465302.modulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT
      StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) A
    FROM `modulabs-project-465302.modulabs_project.data`
  )
  WHERE number_count <= 1
)
--행 1,915개가 삭제됨

-- Description 살펴보기
-- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력
SELECT
  Description,
  COUNT(*) AS frequency
FROM `modulabs-project-465302.modulabs_project.data`
GROUP BY Description
ORDER BY frequency DESC
LIMIT 30
```

```
-- 대소문자가 혼합된 Description이 있는지 확인
SELECT DISTINCT Description
FROM `modulabs-project-465302.modulabs_project.data`
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
--  총 19개의 Description이 대소문자를 혼합하고 있음

-- 서비스 관련 정보를 포함하는 행들을 제거
DELETE
FROM `modulabs-project-465302.modulabs_project.data`
WHERE
  UPPER(Description) LIKE '%POSTAGE%' OR
  UPPER(Description) LIKE '%CARRIAGE%' OR
  UPPER(Description) LIKE '%BANK CHARGES%' OR
  UPPER(Description) LIKE '%ADJUST%' OR
  UPPER(Description) LIKE '%MANUAL%' OR
  UPPER(Description) LIKE '%CHECK%' OR
  UPPER(Description) LIKE '%SAMPLES%' OR
  UPPER(Description) LIKE '%GIFT%'
  -- 행 4,202개가 삭제됨

-- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.data`
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `modulabs-project-465302.modulabs_project.data`

-- UnitPrice 살펴보기
-- UnitPrice에서 이상치 찾기

-- UnitPrice의 최솟값, 최댓값, 평균
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  ROUND(AVG(UnitPrice), 2) AS avg_price
FROM `modulabs-project-465302.modulabs_project.data`
```

```
-- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균
SELECT
  COUNT(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  ROUND(AVG(Quantity), 2) AS avg_quantity
FROM `modulabs-project-465302.modulabs_project.data`
WHERE UnitPrice = 0


--  이 데이터(UnitPrice = 0)를 제거하고 일관된 데이터셋을 유지
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.data`
SELECT *
FROM `modulabs-project-465302.modulabs_project.data`
WHERE UnitPrice > 0

-- RFM 스코어
-- Recency

-- InvoiceDate 컬럼을 연월일 자료형으로 변경
SELECT
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `modulabs-project-465302.modulabs_project.data`

-- 가장 최근 구매 일자를 MAX() 함수로 찾기
SELECT
  MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `modulabs-project-465302.modulabs_project.data`

-- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장
SELECT
    CustomerID,
```

```
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `modulabs-project-465302.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID

-- 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `modulabs-project-465302.modulabs_project.data`
  GROUP BY CustomerID
);

-- 지금까지의 결과를 user_r이라는 이름의 테이블로 저장
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.user_

WITH user_last_purchase AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `modulabs-project-465302.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
),
global_last_purchase AS (
  SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
  FROM `modulabs-project-465302.modulabs_project.data`
)

SELECT
  u.CustomerID,
  u.InvoiceDay,
```

```
  g.most_recent_date,
  DATE_DIFF(g.most_recent_date, u.InvoiceDay, DAY) AS recency
FROM user_last_purchase u
CROSS JOIN global_last_purchase g

-- Frequency
-- 1. 전체 거래 건수 계산
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `modulabs-project-465302.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID

-- 2. 구매한 아이템의 총 수량 계산
-- 각 고객 별로 구매한 아이템의 총 수량을 더해줌
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `modulabs-project-465302.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID

-- '1. 전체 거래 건수 계산'과 '2. 구매한 아이템의 총 수량 계산'의 결과를 합쳐서 user_rf라는
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.user_

WITH purchase_count AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `modulabs-project-465302.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
),

item_quantity AS (
```

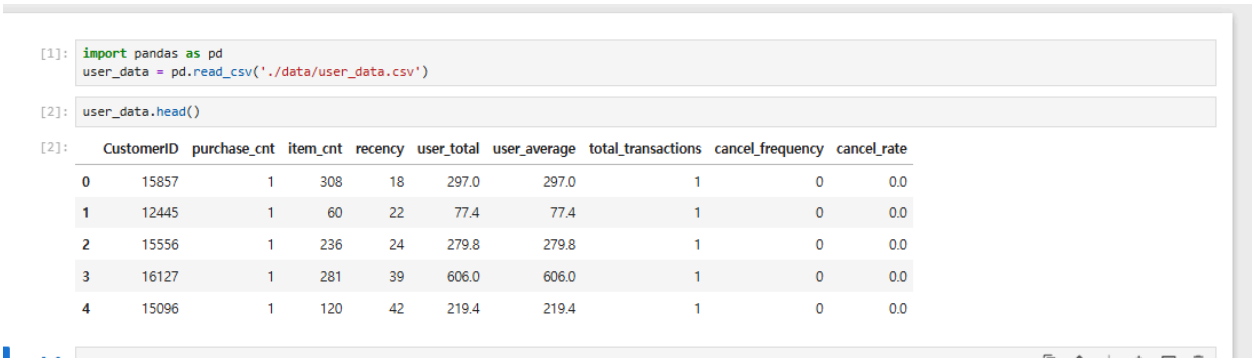
```
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM `modulabs-project-465302.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  iq.item_cnt
FROM purchase_count pc
JOIN item_quantity iq
  ON pc.CustomerID = iq.CustomerID



-- Monetary
-- 1. 고객별 총 지출액 계산
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `modulabs-project-465302.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID

-- 2. 고객별 평균 거래 금액 계산
-- 고객별 평균 거래 금액을 구하기 위해 1) data 테이블을 user_rf 테이블과 조인(LEFT JOI
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.user_
WITH user_base AS (
  SELECT
    r.CustomerID,
    r.recency,
    rf.purchase_cnt,
    rf.item_cnt
  FROM `modulabs-project-465302.modulabs_project.user_r` r
```

```
  JOIN `modulabs-project-465302.modulabs_project.user_rf` rf
  ON r.CustomerID = rf.CustomerID
),
user_total AS (
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM `modulabs-project-465302.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT
  ub.CustomerID,
  ub.purchase_cnt,
  ub.item_cnt,
  ub.recency,
  ut.user_total,
  ROUND(ut.user_total / ub.purchase_cnt, 1) AS user_average
FROM user_base ub
LEFT JOIN user_total ut
ON ub.CustomerID = ut.CustomerID

-- RFM 통합 테이블 출력하기
SELECT *
FROM `modulabs-project-465302.modulabs_project.user_rfm`

-- RFM 고유한 유저의 수
SELECT COUNT(*) AS row_count
FROM `modulabs-project-465302.modulabs_project.user_rfm`



-- 추가 Feature 추출
-- 구매하는 제품의 다양성

-- 1) 고객 별로 구매한 상품들의 고유한 수를 계산합니다. 높은 숫자가 나오는 것은 해당 고객
```

```
-- 이후 2) user_rfm 테이블과 결과를 합치고, 이를 3) user_data라는 이름의 테이블에 저장
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.user_
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `modulabs-project-465302.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT
  ur.*,
  up.unique_products
FROM `modulabs-project-465302.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID

-- 평균 구매 주기
-- 평균 구매 소요 일수를 계산하고, 그 결과를 user_data에 통합
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.user_
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(DATE(InvoiceDate),
                LAG(DATE(InvoiceDate)) OVER (PARTITION BY CustomerID ORDER BY
                DAY) AS interval_
```

```
    FROM `modulabs-project-465302.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT
  u.*,
  pi.average_interval
FROM `modulabs-project-465302.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID

-- 구매 취소 경향성
-- 취소 빈도와 취소 비율을 계산하고 그 결과를 user_data에 통합
CREATE OR REPLACE TABLE `modulabs-project-465302.modulabs_project.user_

WITH user_rfm AS (
  SELECT *
  FROM `modulabs-project-465302.modulabs_project.user_rfm`
),
TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS
  FROM `modulabs-project-465302.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.total_transactions,
  t.cancel_frequency,
  ROUND(t.cancel_frequency / t.total_transactions * 100, 2) AS cancel_rate
```

```
FROM user_rfm u
LEFT JOIN TransactionInfo t
ON u.CustomerID = t.CustomerID

-- 최종적으로 user_data를 출력
SELECT *
FROM `modulabs-project-465302.modulabs_project.user_data`
```



[1]:
```
import pandas as pd
user_data = pd.read_csv('./data/user_data.csv')
```

[2]:
```
user_data.head()
```

[2]:

	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	total_transactions	cancel_frequency	cancel_rate
0	15857	1	308	18	297.0	297.0	1	0	0.0
1	12445	1	60	22	77.4	77.4	1	0	0.0
2	15556	1	236	24	279.8	279.8	1	0	0.0
3	16127	1	281	39	606.0	606.0	1	0	0.0
4	15096	1	120	42	219.4	219.4	1	0	0.0