

고객을 세그먼테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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_InvoiceData,
COUNT(UnitPrice) AS COUNT_UnitPrice,
COUNT(CustomerID) AS COUNT_CustomerID,
COUNT(Country) AS COUNT_Country
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



5-4. 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_percent
FROM 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_percent
FROM 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_perce
FROM 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_percenta
FROM 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_perce
FROM 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_percent
FROM 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_percen
FROM modulabs_project.data
UNION ALL
SELECT
    'Country' AS column_name,
```

 $ROUND(SUM(CASE\ WHEN\ Country\ IS\ NULL\ THEN\ 1\ ELSE\ 0\ END)\ /\ COUNT(*)\ *\ 100,\ 2)\ AS\ missing_percentag\\ FROM\ modulabs_project.data$

[결과 이미지를 넣어주세요]



결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM modulabs_project.data
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]



결측치 처리

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

5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT COUNT(*)
FROM (
   SELECT *, COUNT(*) AS DupleCount
FROM modulabs_project.data
   GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
   HAVING DupleCount > 1
);
```

[결과 이미지를 넣어주세요]



중복값 처리

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

```
CREATE OR REPLACE TABLE `modulabs_project.data` AS

SELECT DISTINCT *

FROM `modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



5-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM modulabs_project.data;
```

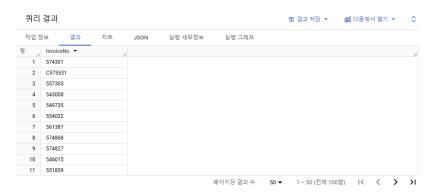
[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM modulabs_project.data
LIMIT 100;
```

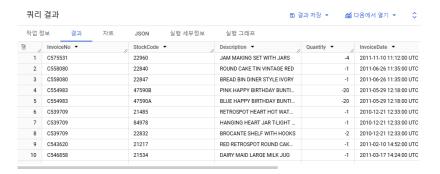
[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM 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 SUBSTR(InvoiceNo, 1, 1) = 'C' THEN 1 ELSE 0 END) / COUNT(*) * 100, 1) AS FROM modulabs_project.data;

[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode)
FROM 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 modulabs_project.data GROUP BY StockCode

```
ORDER BY sell_cnt DESC
LIMIT 10;
```



• StockCode 의 문자열 내 숫자의 길이를 구해보기

```
WITH UniqueStockCodes AS (
    SELECT DISTINCT StockCode
    FROM project_name.modulabs_project.data
)

SELECT
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
    COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC;
```

[결과 이미지를 넣어주세요]

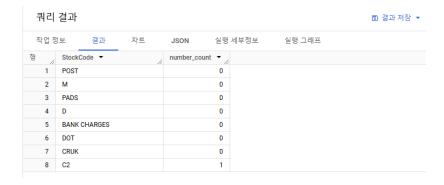


• StockCode 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고

。 **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 modulabs_project.data
)
WHERE number_count IN (0, 1);
```

[결과 이미지를 넣어주세요]



- 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)



• 제품과 관련되지 않은 거래 기록을 제거하기

[결과 이미지를 넣어주세요]

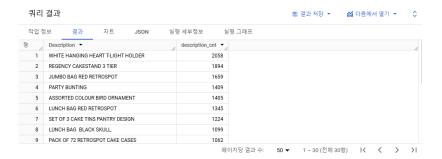


Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

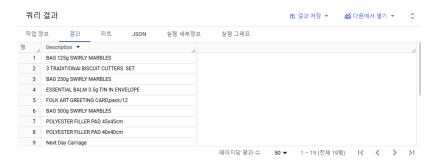
[결과 이미지를 넣어주세요]



• 대소문자가 혼합된 Description이 있는지 확인하기

```
SELECT DISTINCT Description
FROM project_name.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

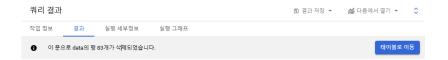
[결과 이미지를 넣어주세요]



• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM modulabs_project.data
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```

[결과 이미지를 넣어주세요]



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기



UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, ROUND(AVG(UnitPrice), 3) AS avg_pri FROM 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
FROM modulabs_project.data
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT *
```



5-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
-- version 1

SELECT
  (SELECT DATE(MAX(InvoiceDate)) FROM modulabs_project.data) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *

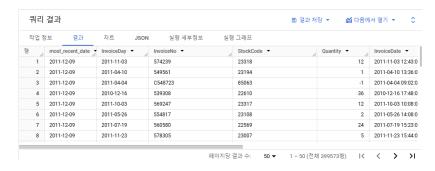
FROM modulabs_project.data;

-- version 2

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

FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
```

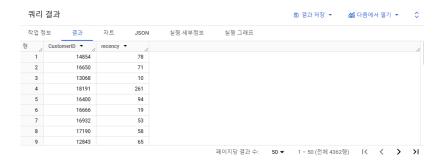
```
FROM modulabs_project.data
GROUP BY CustomerID;
```



• 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 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 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 modulabs_project.data
    GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

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

[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT CustomerID,
SUM(Quantity) AS item_cnt
FROM modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
   SELECT CustomerID,
        COUNT(DISTINCT InvoiceNo) AS purchase_cnt
   FROM modulabs_project.data
   GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
```

```
item_cnt AS (
  SELECT CustomerID,
   SUM(Quantity) AS item_cnt
  FROM 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 modulabs_project.user_r AS ur
    ON pc.CustomerID = ur.CustomerID;
```



Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

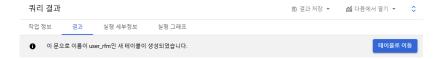
```
SELECT CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
FROM modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]



- 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 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 modulabs_project.user_rf AS rf
 LEFT JOIN (
   SELECT CustomerID,
     ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
   FROM modulabs_project.data
   GROUP BY CustomerID
 ) AS ut
 ON rf.CustomerID = ut.CustomerID;
```



RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

```
SELECT * FROM modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]



5-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

• 1) 고객 별로 구매한 상품들의 고유한 수를 계산하기

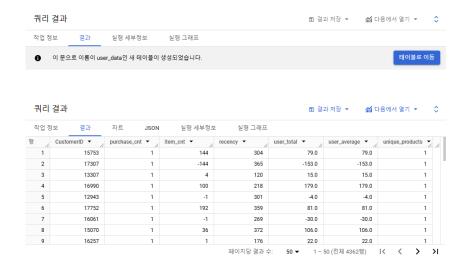
1) 고객 글로 구매한 영남글의 고유한 구글 2) user_rfm 테이블과 결과를 합치기 3) user_data 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE modulabs_project.user_data AS

WITH unique_products AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT StockCode) AS unique_products
        FROM modulabs_project.data
        GROUP BY CustomerID
)

SELECT ur.*, up.* EXCEPT (CustomerID)

FROM 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 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_inte
FROM (
-- (1) 구매와 구매 사이에 소요된 일수
SELECT
CustomerID,
DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
FROM
modulabs_project.data
WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM modulabs_project.user_data AS u
```

```
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```



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

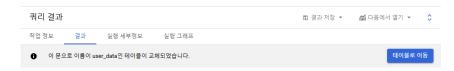
 취소 빈도와 취소 비율을 계산하고 그 결과를 user_data 에 통합하기 (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE modulabs_project.user_data AS

WITH TransactionInfo AS (
   SELECT CustomerID,
   COUNT(InvoiceNo) AS total_transactions,
   SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) AS cancel_frequency
   FROM modulabs_project.data
   GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT (CustomerID),
   ROUND(t.cancel_frequency / t.total_transactions * 100, 2) AS cancel_rate
FROM modulabs_project.user_data AS u
   LEFT JOIN TransactionInfo AS t
   ON u.CustomerID = t.CustomerID;</pre>
```

[결과 이미지를 넣어주세요]



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

```
SELECT * FROM modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]



인사이트 / 회고 정리

인사이트

- 결측치가 전체 데이터에서 차지 하는 비율이 어느 정도이냐에 따라서 결측치 처리 전략이 달라질 수 있을 듯 합니다.
- 구하고자 하는 것이 무엇인지를 명확하게 정의하는 것이 필요하다고 생각합니다.
- 거래 건수가 1개이면서 구매한 아이템 수량이나 총 지출액 등이 (-)인 데이터가 일부 있습니다. 이이런 특이 데이터는 실제 분석 과정에서 체크가 되어 야 할 듯 합니다.
- 구매 건수, 구매액 등과 같은 지표 이외에 제품 다양성, 평균 구매 주기, 구매 취소 경험 등과 같은 다양한 지표이 있다는 것을 알았습니다.

회고

- 실제 데이터를 가지고 프로젝트 형식으로 sql을 공부하니까 구문을 익히고, 이해하는데 도움이 많이 되었습니다.
- 방대한 양의 데이터를 다루기 위해서는 꼼꼼함이 필요할 듯
- 최종 작업을 마치고, 프로젝트 제출 단계에서 BigQuery에서 데이터를 다운로드 받을 때 실수 아닌 실수로 500줄 가까이 작성한 쿼리 파일이 날라갔습니다.. 쿼리 작성할 때도 그 때 그 때 저장을 꼭 해야 겠다는 것을 느꼈습니다. (노션 pdf 파일이 있어서 그나마 다행...)

.