

고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

SELECT * FROM `modulabs_project.data` LIMIT 10



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT COUNT(*) FROM '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,
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.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 missing_value,

ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage FROM `modulabs_project.data`

UNION ALL

SELECT

'stockCode' AS missing_value,

ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage FROM `modulabs_project.data`

UNION ALL

SELECT

'Description' AS missing_value,

ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage FROM `modulabs_project.data`

UNION ALL

SELECT

'Quantity' AS missing_value,

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

UNION ALL

SELECT

'InvoiceDate' AS missing_value,

ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage FROM `modulabs_project.data`

UNION ALL

SELECT

'UnitPrice' AS missing_value,

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

UNION ALL

SELECT

'CustomerID' AS missing_value,

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

UNION ALL

SELECT

'Country' AS missing_value,

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

결측치 처리 전략

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

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



결측치 처리

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

DELETE FROM `modulabs_project.data`
WHERE Description IS NULL
OR Description = '?';

DELETE FROM `modulabs_project.data`
WHERE CustomerID IS NULL

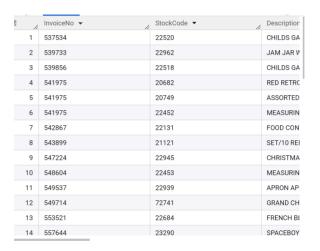


11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

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



중복값 처리

- 중복값을 제거하는 쿼리문 작성하기
 - $\,\circ\,\,$ 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
```

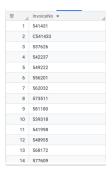


11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT
COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM
modulabs_project.data



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo FROM modulabs_project.data LIMIT 100



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

SELECT *
FROM project_name.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) * 100.0) / COUNT(*),
1
) AS canceled_ratio_percent
FROM
project_name.modulabs_project.data
```



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT
COUNT(DISTINCT StockCode) AS unique_invoice_count
FROM
modulabs_project.data



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

SELECT StockCode, COUNT(*) AS sell_cnt FROM project_name.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 modulabs_project.data
)
WHERE number_count <= 1;
```

실수로 빼먹어서 지워진 값 → 없는 데이터로 나와요..



- 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 number_count <= 1 THEN 1 ELSE 0 END) * 100.0) / COUNT(*),
2
) AS ratio_of_low_digit_codes
FROM
(
SELECT
StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM
modulabs_project.data
) AS subquery;
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM 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

modulabs_project.data
)

WHERE number_count <= 1
);
```



Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기



• 서비스 관련 정보를 포함하는 행들을 제거하기



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT
* EXCEPT (Description),
UPPER(Description) AS Description
FROM modulabs_project.data

작업 정보 결과 실행 세부정보 실행 그래프

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 project_name.modulabs_project.data;
```

< 작업정보 결과 시각화 JSON 설립세부증 전 min_price * max_price * wng_price * 1 0.0 649.5 2.5046959757405... • 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

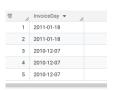


11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT Date(InvoiceDate) As InvoiceDay FROM project_name.modulabs_project.data;



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT

MAX(InvoiceDate) OVER () AS most_recent_date,

DATE(InvoiceDate) AS InvoiceDay,

*

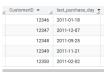
FROM

modulabs_project.data;
```



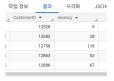
• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS last_purchase_day
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
);
```



• 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,

DATE_DIFF(

(SELECT MAX(DATE(InvoiceDate)) FROM modulabs_project.data),

MAX(DATE(InvoiceDate)),

DAY

) AS recency

FROM

modulabs_project.data

WHERE

CustomerID IS NOT NULL
```

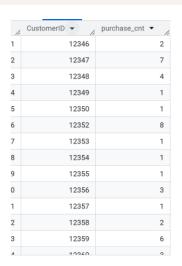
GROUP BY CustomerID;



Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

SELECT
CustomerID,
COUNT(InvoiceNO) AS purchase_cnt
FROM project_name.modulabs_project.data
GROUP BY CustomerID;



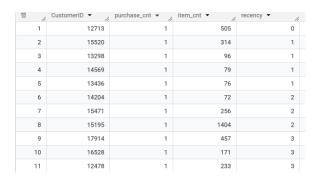
• 각 고객 별로 구매한 아이템의 총 수량 더하기

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

CustomerID	· //	item_cnt ▼	//
	12346		1
	12347		103
	12348		21
	12349		72
	12350		16
	12352		57
	12353		4
	12354		58
	12355		13
	12356		52
	12357		131
	12358		12
	12359		214

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

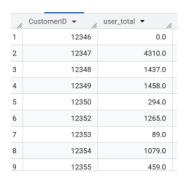
```
CREATE OR REPLACE TABLE modulabs_project.user_rf AS
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM
    modulabs_project.data
  WHERE
    CustomerID IS NOT NULL
  GROUP BY
    CustomerID
),
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM
    modulabs_project.data
  WHERE
    CustomerID IS NOT NULL
  GROUP BY
    CustomerID
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.recency
FROM
  purchase_cnt AS pc
INNER JOIN
  item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
INNER 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(Quantity * UnitPrice)) AS user_total
FROM
modulabs_project.data
WHERE
CustomerID IS NOT NULL
GROUP BY
CustomerID;
```



- 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,
  ROUND(ut.user_total / rf.purchase_cnt) AS user_average
FROM
  modulabs_project.user_rf rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM
    modulabs_project.data
```

```
WHERE
CustomerID IS NOT NULL
GROUP BY
CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

select * from `modulabs_project.user_rfm`

Cust	omerID ▼ / purch	ase_cnt ▼ / item_	cnt ▼ recency	v user.	_total ▼user	_average ▼
1	12713	1	505	0	795.0	795.0
2	13298	1	96	1	360.0	360.0
3	13436	1	76	1	197.0	197.0
4	14569	1	79	1	227.0	227.0
5	15520	1	314	1	343.0	343.0
6	14204	1	72	2	151.0	151.0
7	15471	1	256	2	454.0	454.0
8	15195	1	1404	2	3861.0	3861.0

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 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;

작업 정보 결과 실행 세

작업 정보 결과 실행 세부정보 실행 그래프

● 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 project_name.modulabs_project.user_data AS
WITH purchase_intervals AS (
SELECT
 CustomerID,
 CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
 FROM (
 SELECT
  CustomerID,
  DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
  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;
```

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 modulabs_project.user_data AS

WITH TransactionInfo AS (
    SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
FROM
    modulabs_project.data
WHERE
    CustomerID IS NOT NULL
GROUP BY
    CustomerID
)

SELECT
    u.*,
```

t.* EXCEPT(CustomerID),
ROUND((t.cancel_frequency * 100.0) / t.total_transactions, 2) AS cancel_rate
FROM
modulabs_project.user_data AS u
LEFT JOIN
TransactionInfo AS t
ON
u.CustomerID = t.CustomerID;

작업 정보 결과 실행 세부정보 실행 그래프

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

select * from modulabs_project.user_data



회고

[회고 내용을 작성해주세요]

Keep : 오늘 문제가 발생해도 계속 원인을 찾으려고 하고 결국에는 문제를 잘 수정했다는 것! 앞으로도 잘 했으면 좋겠다. 이전시간에 배운 내용들을 많이 기억해서 잘 사용하려고 해서 잘 문제를 풀어낼 수 있었다. (아닌 것들도 있었음) 그치만 이전 내용들 컨닝하면서 많이 쓰려고 한 점 굿굿. 데이터 정제 부분은 정말 쉽게 넘어와서 다행이다.

Problem : 중간에 frequency 부분에서 오류가 나고 멘붕이 와서 monetary에서 완전 바보같은 행동을 한 점은 반성해야한다. 제정신이 아닌 채로 쿼리를 짜니깐 실수가 눈에 들어오지 않고 다른 곳으로 신경이 쏠려서 제대로된 문제점을 파악할 수 없었다. 정신을 차리고 다시 보니깐 눈에 들어오긴 했지만 (퍼실님의 한마디에..) 앞으로는 이런 일이 없었으면 좋겠다. 시간을 너무 잡아먹었다.

Try : 어제 조퇴를 해서 어제 노드학습을 제대로 하지 않은 상태에서 오늘 노드를 하기 시작했다. 어제 노드를 완벽하게 숙지하고 오늘 것을 했으면 더 수 월하게 진행했을 것 같다. 앞으로는 전 노드를 잘 확인하고 예습복습 철저히!!!하자..(은별님이 복습을 아주 열심히하더군요..)