# 11 - 미니프로젝트 - 고객세그멘테이션

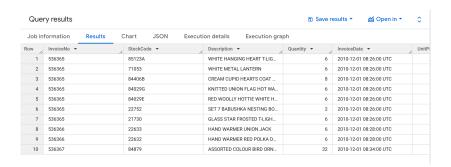
https://www.kaggle.com/datasets/carrie1/ecommerce-data

# 11-2. 데이터 불러오기

#### 데이터 살펴보기

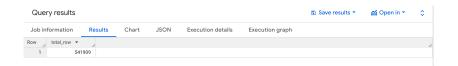
• 테이블에 있는 10개의 행만 출력하기

select \* from poetic-glass-456004-g2.modulabs\_project.data limit 10;



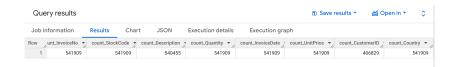
• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

select count(\*) from poetic-glass-456004-g2.modulabs\_project.data;



#### 데이터 수 세기

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

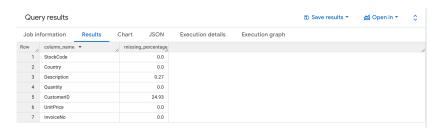


# 11-4. 데이터 전처리 방법(1): 결측치 제거

#### 컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계산
  - $\circ$  각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

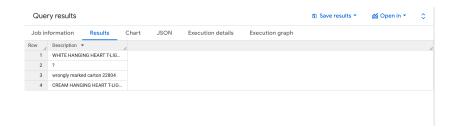
```
SELECT
  'InvoiceNo' AS column name.
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.modulabs_project.data
union all
SELECT
  'StockCode' AS column_name,
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.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 poetic-glass-456004-g2.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 poetic-glass-456004-g2.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 poetic-glass-456004-g2.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 poetic-glass-456004-g2.modulabs_project.data
SELECT
  'Country' AS column_name,
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.modulabs_project.data;
```



### 결측치 처리 전략

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

```
select distinct Description
from poetic-glass-456004-g2.modulabs_project.data
where StockCode = '85123A';
```



## 결측치 처리

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

delete from poetic-glass-456004-g2.modulabs\_project.data where Description is null or 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

 $select\ count(*)\ duplicate\ from\ poetic-glass-456004-g2. modulabs\_project. data \\ group\ by\ InvoiceNo,\ StockCode,\ Description,\ Quantity,\ InvoiceDate,\ UnitPrice,\ CustomerID,\ Country\ having\ count(*)\ >\ 1;$ 



# 중복값 처리

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

create or replace table poetic-glass-456004-g2.modulabs\_project.data as select distinct \* from poetic-glass-456004-g2.modulabs\_project.data;



# 11-6. 데이터 전처리(3): 오류값 처리

# InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

select count(distinct InvoiceNo) as count\_unique\_invNo from poetic-glass-456004-g2.modulabs\_project.data;



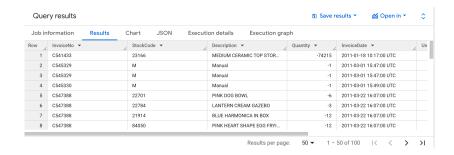
• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

-- 고유 invoice list 100 select distinct InvoiceNo from poetic-glass-456004-g2.modulabs\_project.data order by InvoiceNo limit 100;



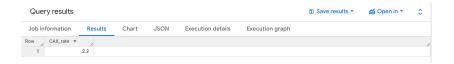
• InvoiceNo 가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)

select \* from poetic-glass-456004-g2.modulabs\_project.data where InvoiceNo like 'C%' or InvoiceNo like 'c%' limit 100;



• 구매 건 상태가 Canceled 인 데이터의 비율(%) - 소수점 첫번째 자리까지

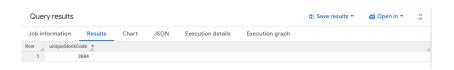
select round(sum(case when InvoiceNo like 'C%' then 1 when InvoiceNo like 'c%' then 1 else 0 end) \* 100 / count(\*), 1) as CAX\_from poetic-glass-456004-g2.modulabs\_project.data;



#### StockCode 살펴보기

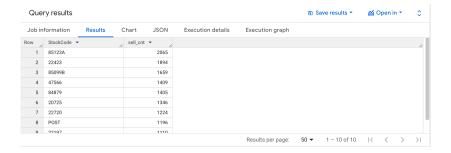
• 고유한 StockCode 의 개수를 출력하기

select count(distinct StockCode) uniqueStockCode from poetic-glass-456004-g2.modulabs\_project.data;



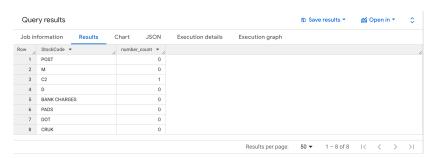
- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기
  - 。 상위 10개의 제품들을 출력하기

select StockCode, count(\*) sell\_cnt from poetic-glass-456004-g2.modulabs\_project.data group by StockCode order by sell\_cnt desc limit 10;



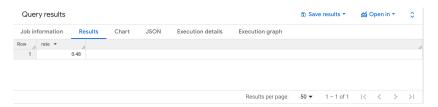
- StockCode 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
  - **숫자가 0~1개인 값**들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT StockCode, number_count
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM poetic-glass-456004-g2.modulabs_project.data
)
where number_count in (0,1);
```



- 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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
SELECT round(cast(countif(number_count in (0,1)) as int64)/ count(*) * 100 , 2) rate
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM poetic-glass-456004-g2.modulabs_project.data);
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
delete from poetic-glass-456004-g2.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 poetic-glass-456004-g2.modulabs_project.data
)
where number_count in (0,1)
);
```

 Query results
 © Save results ▼
 © Open in ▼
 ♦

 Job information
 Results
 Execution details
 Execution graph

 This statement removed 1,915 rows from data.
 Go to table

11 - 미니프로젝트 - 고객세그멘테이션 6

## Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

select Description,count(\*) cnt from poetic-glass-456004-g2.modulabs\_project.data group by Description order by cnt desc limit 30;



• 서비스 관련 정보를 포함하는 행들을 제거하기

delete from poetic-glass-456004-g2.modulabs\_project.data where Description like '%High Resolution Image%' or Description like '%Next Day Carriage%';



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs\_project.data AS SELECT \* EXCEPT (Description), upper(Description) Description FROM poetic-glass-456004-g2.modulabs\_project.data;



#### UnitPrice 살펴보기

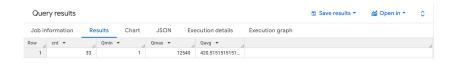
• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

select min(UnitPrice) min, max(UnitPrice) max, avg(UnitPrice) avg from poetic-glass-456004-g2.modulabs\_project.data;



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

 $select\ count(InvoiceNo)\ cnt,\ min(Quantity)\ Qmin,\ max(Quantity)\ Qmax,\ avg(Quantity)Qavg\ from\ poetic-glass-456004-g2.modulabs\_project.data$  where UnitPrice = 0;



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs\_project.data AS SELECT \*
FROM poetic-glass-456004-g2.modulabs\_project.data
WHERE UnitPrice != 0;

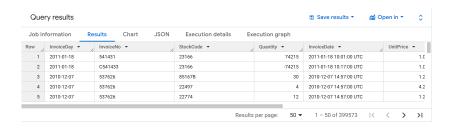


# 11-7. RFM 스코어

#### Recency

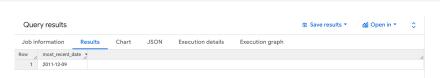
• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

select date(InvoiceDate) as InvoiceDay, \* from poetic-glass-456004-g2.modulabs\_project.data;



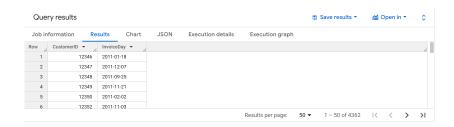
• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

select date(max(InvoiceDate)) most\_recent\_date from poetic-glass-456004-g2.modulabs\_project.data;



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

select CustomerID, Date(Max(invoiceDate)) as InvoiceDay from poetic-glass-456004-g2.modulabs\_project.data group by CustomerID;

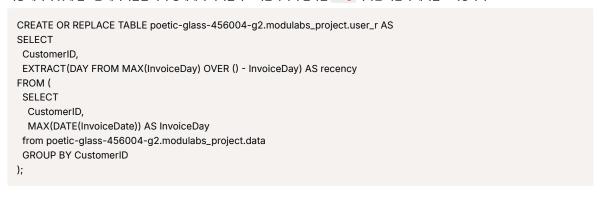


• 가장 최근 일자( most\_recent\_date )와 유저별 마지막 구매일( InvoiceDay )간의 차이를 계산하기

```
SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
from poetic-glass-456004-g2.modulabs_project.data
GROUP BY CustomerID
);
```



• 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(InvoiceNo) as purchase\_cnt from poetic-glass-456004-g2.modulabs\_project.data group by CustomerID;



• 각 고객 별로 구매한 아이템의 총 수량 더하기

select CustomerID, count(InvoiceNo) as purchase\_cnt from poetic-glass-456004-g2.modulabs\_project.data group by CustomerID;



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

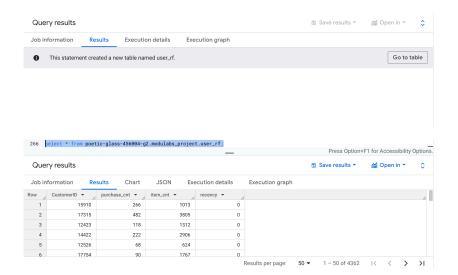
```
CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs_project.user_rf AS
WITH purchase_cnt AS (
select CustomerID, count(InvoiceNo) as purchase_cnt
from poetic-glass-456004-g2.modulabs_project.data
group by CustomerID), -- (1) 전체 거래 건수 계산
item_cnt AS (
select CustomerID, sum(Quantity) item_cnt
from poetic-glass-456004-g2.modulabs_project.data
group by CustomerID) -- (2) 구매한 아이템 총 수량 계산
SELECT
               -- 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
```

11 - 미니프로젝트 - 고객세그멘테이션 10

ON pc.CustomerID = ic.CustomerID

JOIN poetic-glass-456004-g2.modulabs\_project.user\_r AS ur

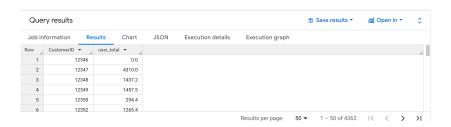
ON pc.CustomerID = ur.CustomerID;



#### **Monetary**

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

select CustomerID , round(SUM(UnitPrice \* Quantity),1) user\_total from poetic-glass-456004-g2.modulabs\_project.data 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 poetic-glass-456004-g2.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) user_average

FROM poetic-glass-456004-g2.modulabs_project.user_rf rf

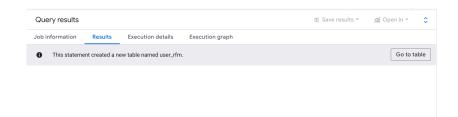
LEFT JOIN (

SELECT CustomerID , round(SUM(UnitPrice * Quantity),1) user_total

from poetic-glass-456004-g2.modulabs_project.data
```

11 - 미니프로젝트 - 고객세그멘테이션 11

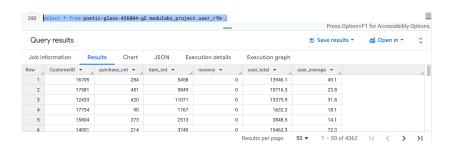
group by CustomerID ) ut ON rf.CustomerID = ut.CustomerID;



#### RFM 통합 테이블 출력하기

• 최종 user\_rfm 테이블을 출력하기

select \* from poetic-glass-456004-g2.modulabs\_project.user\_rfm;



# 11-8. 추가 Feature 추출

## 1. 구매하는 제품의 다양성

1) 고객 별로 구매한 상품들의 고유한 수를 계산하기
 2)

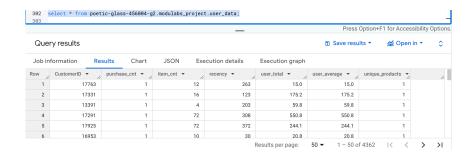
user\_rfm 테이블과 결과를 합치기

3)

user\_data 라는 이름의 테이블에 저장하기

CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs\_project.user\_data AS
WITH unique\_products AS (
SELECT
CustomerID,
COUNT(DISTINCT StockCode) AS unique\_products
FROM poetic-glass-456004-g2.modulabs\_project.data
GROUP BY CustomerID
)
SELECT ur.\*, up.\* EXCEPT (CustomerID)
FROM poetic-glass-456004-g2.modulabs\_project.user\_rfm AS ur
JOIN unique\_products AS up
ON ur.CustomerID = up.CustomerID;

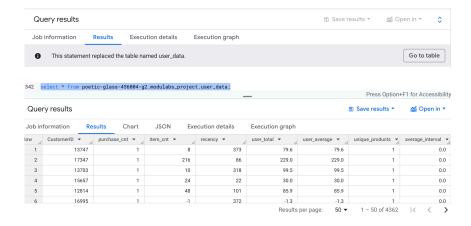




#### 2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
  - 。 균 구매 소요 일수를 계산하고, 그 결과를 user\_data 에 통합

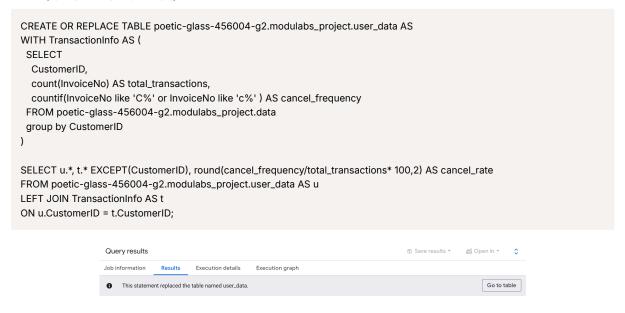
```
CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs_project.user_data AS
WITH purchase_intervals AS (
 -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
SELECT
 CustomerID,
 CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
 FROM (
  -- (1) 구매와 구매 사이에 소요된 일수
  SELECT
   CustomerID,
   DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
   poetic-glass-456004-g2.modulabs_project.data
  WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM poetic-glass-456004-g2.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)



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

select \* from poetic-glass-456004-g2.modulabs\_project.user\_data;



## 회고

[회고 내용을 작성해주세요]

Keep: 스스로 생각해서 문제를 해결하려고 했다.

Problem : 자주 나오는 것임에도 코드를 잘 외우고 있지는 않은 거 같다 (근데, 모든 코드를 암기하는게 효율적인가? 의구심)

Try : 프로그래밍을 하다보면 활용해야 하는 요구하는 사항이 비슷한데, 자주 쓰이는 코드는 옵시디언 같은 걸로 정리해서 각 프로그래밍별로 비교 가능하게 정리를 해 둬서 빨리 찾아 쓸 수 있게 해야겠다...고 계속 생각하고 있는데 못하고 있네요;;;;