

고객을 세그먼테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
select * from amiable-nova-447401-t6.modulabs_project.data limit 10;
```

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
select count(*) from amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



데이터 수 세기

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

```
# [[YOUR QUERY]]
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 amiable-nova-447401-t6.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

```
# [[YOUR QUERY]]
    'InvoiceNo' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM amiable-nova-447401-t6.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 amiable-nova-447401-t6.modulabs_project.data UNION ALL
   'Description' AS column_name,
   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentag
FROM amiable-nova-447401-t6.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 amiable-nova-447401-t6.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_percentag
FROM amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행 //	column_name ▼	missing_percentage
1	UnitPrice	0.0
2	Country	0.0
3	Quantity	0.0
4	InvoiceNo	0.0
5	InvoiceDate	0.0
6	Description	0.27
7	CustomerID	24.93
8	StockCode	0.0

```
//참고_ 위와 결과 같음
SELECT column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(

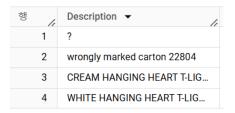
SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM amiable-
SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM amiable-
SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM amiable-
SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM amiable-nc
SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM amiable-
SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM amiable-
SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM amiable
SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM amiable-nova
) AS column_data;
```

결측치 처리 전략

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

```
SELECT # [YOUR QUERY]
FROM project_name.modulabs_project.data
# [YOUR QUERY]; distinct(Description)
```

[결과 이미지를 넣어주세요]



결측치 처리

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

```
DELETE FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]]CustomerID is null or Description is null;
```

[결과 이미지를 넣어주세요]

● 이 문으로 data의 행 135,080개가 삭제되었습니다.

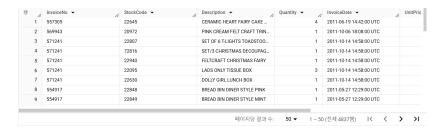
5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT *
FROM project_name.modulabs_project.data
# [[YOUR QUERY]]
select * FROM amiable-nova-447401-t6.modulabs_project.data
group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
having count(*)>1;
```

[결과 이미지를 넣어주세요]



중복값 처리

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

```
# [[YOUR QUERY]];
CREATE OR REPLACE TABLE amiable-nova-447401-t6.modulabs_project.data
as select distinct * from amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



5-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
# [[YOUR QUERY]]
select count(distinct InvoiceNo) from amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
# [[YOUR QUERY]]
select distinct InvoiceNo from amiable-nova-447401-t6.modulabs_project.data limit 100;
```

[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]] InvoiceNo like 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]



• 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 project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

```
# [[YOUR QUERY]]
select count(distinct StockCode) from amiable-nova-447401-t6.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 project_name.modulabs_project.data
# group by StockCode
ORDER BY sell_cnt DESC
# limit 10;
```

[결과 이미지를 넣어주세요]

• 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]

행 //	StockCode ▼	sell_cnt ▼
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
Q	22197	1110

- 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 amiable-nova-447401-t6.modulabs_project.data
)
WHERE # number_count <=1;</pre>
```

행	11	StockCode ▼	number_count ▼
	1	POST	0
	2	М	0
	3	PADS	0
	4	D	0
	5	BANK CHARGES	0
	6	DOT	0
	7	CRUK	0
	8	C2	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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
# [[YOUR QUERY]]
select ROUND(SUM(CASE WHEN number_count <=1 THEN 1 ELSE 0 END)/count(*)*100, 2)
FROM (
   SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
   FROM amiable-nova-447401-t6.modulabs_project.data
) ;</pre>
```

[결과 이미지를 넣어주세요]



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM amiable-nova-447401-t6.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 amiable-nova-447401-t6.modulabs_project.data) WHERE number_count <=1
);
```

[결과 이미지를 넣어주세요]

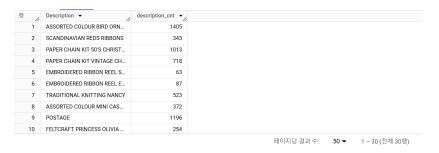


Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM project_name.modulabs_project.data
# group by Description 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



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE amiable-nova-447401-t6.modulabs_project.data AS

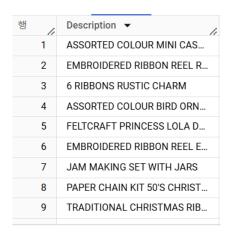
SELECT

* EXCEPT (Description),

# upper(Description) AS Description

FROM amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

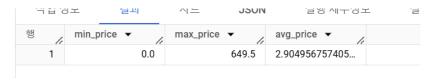


UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

SELECT # min(UnitPrice) AS min_price, max(UnitPrice) AS max_price, avg(UnitPrice) AS avg_price FROM project_name.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, a FROM project_name.modulabs_project.data
WHERE # UnitPrice= 0;

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

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

SELECT *

FROM project_name.modulabs_project.data

WHERE UnitPrice != 0;
```

[결과 이미지를 넣어주세요]

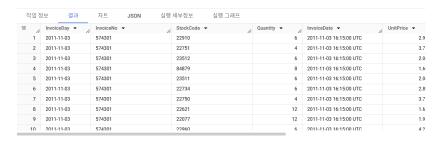


5-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT # Date(InvoiceDate) AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

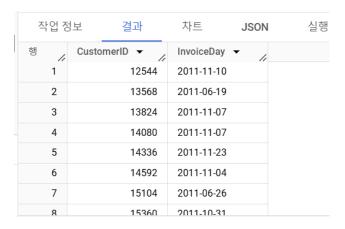
```
SELECT
  # MAX(Date(InvoiceDate)) AS most_recent_date,
  # Date(InvoiceDate) AS InvoiceDay,
  *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
# MAX(Date(InvoiceDate)) AS InvoiceDay
FROM project_name.modulabs_project.data
# group by CustomerID;
```

[결과 이미지를 넣어주세요]



• 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
```

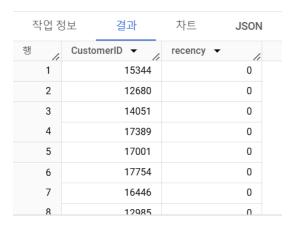
```
FROM project_name.modulabs_project.data
GROUP BY CustomerID
);
```

작업 정	J보 결과	차트	JSON
행 //	CustomerID ▼	recency 🕶	11
1	15360		39
2	13314		1
3	13061		72
4	15373		8
5	13328		316
6	12822		70
7	17942		7
8	12847		22
9	16689		75

• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user_r 이라는 이름의 테이블로 저장하기

```
CREATE OR REPLACE TABLE project_name.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 amiable-nova-447401-t6.modulabs_project.data2
   GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
# count(InvoiceNo) AS purchase_cnt
FROM project_name.modulabs_project.data
#group by CustomerID;
```

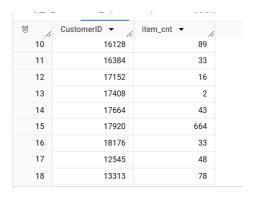
[결과 이미지를 넣어주세요]

작업 정	경보 결과	차트	JSON
행 //	CustomerID ▼	purchase_cnt	· /
1	12544		19
2	13568		43
3	13824		46
4	14080		4
5	14336		90
,	14500		150

• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
# count(Quantity) AS item_cnt
FROM project_name.modulabs_project.data
# group by CustomerID;
```

[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
#
SELECT
```

```
CustomerID,
 count(InvoiceNo) AS purchase_cnt
{\tt FROM\ amiable-nova-447401-t6.modulabs\_project.data2}
group by CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
 SELECT
 CustomerID,
 count(Quantity) AS item_cnt
FROM amiable-nova-447401-t6.modulabs_project.data2
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 project_name.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)) AS user_total
FROM project_name.modulabs_project.data
# group by CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID ▼	user_total ▼
1	12544	54.0
2	13568	131.0
3	13824	127.0
4	14080	4.0
5	14336	145.0
6	14592	324.0
7	15104	365.0
8	15360	31.0

- 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 테이블 로 저장하기

```
CREATE OR REPLACE TABLE project_name.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 project_name.modulabs_project.user_rf rf
LEFT JOIN (
 -- 고객 별 총 지출액
 SELECT
  SELECT
   CustomerID,
   Round(sum(UnitPrice)) AS user_total
 FROM amiable-nova-447401-t6.modulabs_project.data2
 GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

● 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

```
# select * from amiable-nova-447401-t6.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

행 //	CustomerID ▼	purchase_cnt ▼	item_cnt ▼	recency ▼	user_total ▼	user_average ▼ //
1	15694	78	78	0	357.0	4.576923076923
2	14422	222	222	0	624.0	2.810810810810
3	12985	78	78	0	146.0	1.871794871794
4	12423	118	118	0	245.0	2.076271186440
5	12526	68	68	0	184.0	2.705882352941
6	17581	451	451	0	1599.0	3.545454545454
7	16705	284	284	0	1284.0	4.521126760563
8	17364	409	409	0	1115.0	2.726161369193

5-8. 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 project_name.modulabs_project.user_data AS
WITH unique_products AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT StockCode) AS unique_products
    FROM project_name.modulabs_project.data
        GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]



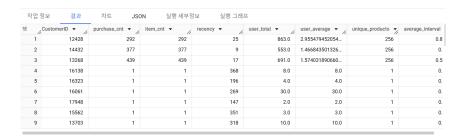
행 //	CustomerID ▼	purchase_cnt ▼ //	item_cnt ▼	recency ▼	user_total ▼	user_average ▼ //	unique_products 🏏
1	12428	292	292	25	863.0	2.955479452054	256
2	14432	377	377	9	553.0	1.466843501326	256
3	13268	439	439	17	691.0	1.574031890660	256
4	16428	1	1	81	3.0	3.0	1
5	15668	1	1	217	1.0	1.0	1
6	13366	1	1	50	0.0	0.0	1
7	16953	1	1	30	2.0	2.0	1
8	16737	1	1	53	1.0	1.0	1
9	13829	1	1	359	9.0	9.0	1
10	13135	1	1	196	1.0	1.0	1

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 (
```

```
-- (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
   -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS
     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;
```



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)

[결과 이미지를 넣어주세요]

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

select * from amiable-nova-447401-t6.modulabs_project.user_data;;

[결과 이미지를 넣어주세요]

