고객을 세그먼테이션하자 [프로젝트]

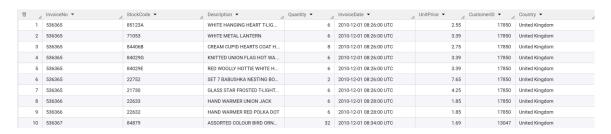
11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM enhanced-ward-466622-g5.1st_project.Data1
LIMIT 10;
```

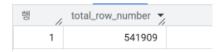
[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT count(*) AS total_row_number
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[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_InvoiceDate,
count(UnitPrice) AS UnitPrice,
count(CustomerID) AS COUNT_CustomerID,
count(Country) AS COUNT_Country
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]



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 enhanced-ward-466622-g5.1st_project.Data1

UNION ALL

SELECT

'StockCode' AS column_name,

 $ROUND(SUM(CASE\ WHEN\ StockCode\ IS\ NULL\ THEN\ 1\ ELSE\ 0\ END)\ /\ COUNT(*)\ *\ 100,\ 2)\ AS\ missing_percentage\\ FROM\ enhanced-ward-466622-g5.1st_project.Data1$

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 enhanced-ward-466622-g5.1st_project.Data1

UNION ALL

SELECT.

'Quantity' AS column_name,

 $ROUND (SUM (CASE \ WHEN \ Quantity \ IS \ NULL \ THEN \ 1 \ ELSE \ 0 \ END) \ / \ COUNT(*)*100, \ 2) \ AS \ missing_percentage \\ FROM \ enhanced-ward-466622-g5.1st_project.Data1$

UNION ALL

SELECT

'InvoiceDate' AS column_name,

ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage FROM enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-q5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1;

[결과 이미지를 넣어주세요]

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]



결측치 처리

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

[결과 이미지를 넣어주세요]

DELETE FROM enhanced-ward-466622-g5.1st_project.Data1 WHERE Description IS NULL OR CustomerID IS NULL;

① 이 문으로 Data1의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- 중복된 행의 수를 세어보기
 - $_{\circ}~$ 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

```
WITH DC AS (
SELECT
InvoiceNo, StockCode, Description, Quantity,
InvoiceDate, UnitPrice, CustomerID, Country,
COUNT(*) AS row_count
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY
InvoiceNo, StockCode, Description, Quantity,
InvoiceDate, UnitPrice, CustomerID, Country
)

SELECT COUNT(*) AS `중복 행 갯수`
```



중복값 처리

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

CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.Data1 AS
SELECT DISTINCT *
FROM enhanced-ward-466622-g5.1st_project.Data1;

SELECT count(*) AS new_total_row_number
FROM enhanced-ward-466622-g5.1st_project.Data1;

[결과 이미지를 넣어주세요]



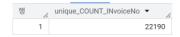
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT count(DISTINCT InvoiceNo) AS unique_COUNT_INvoiceNo FROM enhanced-ward-466622-g5.1st_project.Data1;

[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo FROM enhanced-ward-466622-g5.1st_project.Data1 LIMIT 100;

[결과 이미지를 넣어주세요]



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

SELECT *
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE 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)*100)/COUNT(*),1) AS CancelRatio FROM enhanced-ward-466622-g5.1st_project.Data1;

[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT count(DISTINCT StockCode) AS unique_COUNT_StockCode FROM enhanced-ward-466622-g5.1st_project.Data1;

[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 enhanced-ward-466622-g5.1st_project.Data1
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

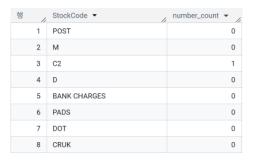
[결과 이미지를 넣어주세요]

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
9	22197		1110
10	23203		1108

- 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 enhanced-ward-466622-g5.1st_project.Data1
)
WHERE number_count <=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)

```
SELECT StockCode, ROUND((COUNT(*) * 100.0) / (SELECT COUNT(*) FROM enhanced-ward-466622-g5.1st_project.Data1), 2) FROM enhanced-ward-466622-g5.1st_project.Data1

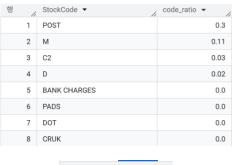
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1

GROUP BY StockCode;

SELECT ROUND((COUNT(*) * 100.0) / (SELECT COUNT(*) FROM enhanced-ward-466622-g5.1st_project.Data1), 2) AS total_coc FROM enhanced-ward-466622-g5.1st_project.Data1

WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1;
```

[결과 이미지를 넣어주세요]





• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM enhanced-ward-466622-g5.1st_project.Data1

WHERE StockCode IN (

SELECT DISTINCT StockCode

FROM (

SELECT StockCode

FROM enhanced-ward-466622-g5.1st_project.Data1

WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1
)
);
```

[결과 이미지를 넣어주세요]

● 이 문으로 Data1의 행 1,915개가 삭제되었습니다.

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT Description, COUNT(*) AS description_cnt FROM enhanced-ward-466622-g5.1st_project.Data1 GROUP BY Description ORDER BY 2 DESC LIMIT 30;

[결과 이미지를 넣어주세요]



• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE

FROM enhanced-ward-466622-g5.1st_project.Data1 WHERE Description IN ('High Resolution Image','Next Day Carriage');

[결과 이미지를 넣어주세요]

이 문으로 Data1의 행 83개가 삭제되었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.Data1 AS SELECT

* EXCEPT (Description), UPPER(Description) AS Description

FROM enhanced-ward-466622-g5.1st_project.Data1;

[결과 이미지를 넣어주세요]

① 이 문으로 이름이 Data1인 테이블이 교체되었습니다.

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 enhanced-ward-466622-g5.1st_project.Data1;

[결과 이미지를 넣어주세요]

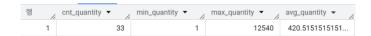


• 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기

SELECT COUNT(Quantity) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS FROM enhanced-ward-466622-g5.1st_project.Data1

WHERE UnitPrice = 0;

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

DELETE FROM enhanced-ward-466622-g5.1st_project.Data1 WHERE UnitPrice = 0;

[결과 이미지를 넣어주세요]

이 문으로 Data1의 행 33개가 삭제되었습니다.

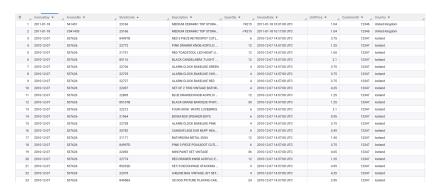
11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay, * FROM enhanced-ward-466622-g5.1st_project.Data1;

[결과 이미지를 넣어주세요]



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT

MAX(DATE(InvoiceDate)) OVER () AS most_recent_date, DATE(InvoiceDate) AS InvoiceDay, * FROM enhanced-ward-466622-g5.1st_project.Data1;

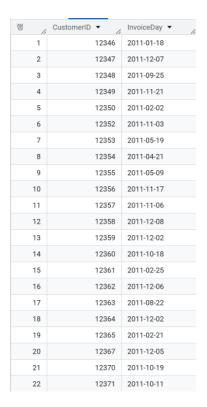
[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT CustomerID, MAX(DATE(InvoiceDate)) AS InvoiceDay FROM enhanced-ward-466622-g5.1st_project.Data1 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 enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

행 //	CustomerID	· //	recency ▼	//
1		12373	3	311
2		12588		39
3		13038		80
4		13135	1	96
5		13499		17
6		13503		71
7		13865		58
8		13924		1
9		13984		26
10		14261		50
11		14411	3	310
12		14462		63
13		14560		7
14		14698		1
15		14747	2	247
16		15004	1	47
17		15158		45
18		15444		9
19		15577	1	51
20		15611		9
21		15900		23
22		16050	1	73
23		16402	2	264
24		16468		74

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

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.user_r AS SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency,
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

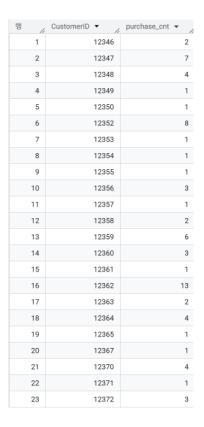
● 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

SELECT
CustomerID,
COUNTt(DISTINCT InvoiceNo) AS purchase_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID;

[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID;

[결과 이미지를 넣어주세요]

행 //	CustomerID ▼	item_cnt ▼
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240
10	12356	1573
11	12357	2708
12	12358	242
13	12359	1599
14	12360	1156
15	12361	90
16	12362	2180
17	12363	408
18	12364	1499
19	12365	173
20	12367	172
21	12370	2349
22	12371	582
23	12372	788

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

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.user_rf AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
CustomerID,
COUNT(Quantity) AS item_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
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 enhanced-ward-466622-g5.1st_project.user_r AS ur
ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

① 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

행 //	CustomerID ▼	purchase_cnt ▼	item_cnt ▼	recency ▼
1	12713	1	505	0
2	15520	1	314	1
3	13298	1	96	1
4	14569	1	79	1
5	13436	1	76	1
6	15195	1	1404	2
7	15471	1	256	2
8	14204	1	72	2
9	12478	1	233	3
10	12650	1	250	3
11	17914	1	457	3
12	14578	1	240	3
13	16569	1	93	3
14	15318	1	642	3
15	16528	1	171	3
16	12442	1	181	3
17	15992	1	17	3
18	12367	1	172	4
19	14219	1	78	4
20	15097	1	170	4
21	17383	1	148	4
22	16597	1	184	4
23	13790	1	748	4

Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

SELECT
CustomerID,
ROUND(SUM(UnitPrice*Quantity)) AS user_total
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID;

[결과 이미지를 넣어주세요]

행 //	CustomerID	• /	user_total	· //
1		12346		0.0
2		12347		4310.0
3		12348		1437.0
4		12349		1458.0
5		12350		294.0
6		12352		1265.0
7		12353		89.0
8		12354		1079.0
9		12355		459.0
10		12356		2487.0
11		12357		6208.0
12		12358		928.0
13		12359		6183.0
14		12360		2302.0
15		12361		175.0
16		12362		4666.0
17		12363		552.0
18		12364		1208.0
19		12365		321.0
20		12367		151.0
21		12370		3422.0
22		12371		1528.0
23		12372		1196.0
24		12373		325.0
25		12374		623.0

• 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 테이블로 저장하기

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_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) AS user_average
FROM enhanced-ward-466622-g5.1st_project.user_rf AS rf
LEFT JOIN (
-- 고객 별 총 지출액
SELECT
 CustomerID,
 ROUND(SUM(UnitPrice*Quantity)) AS user_total
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

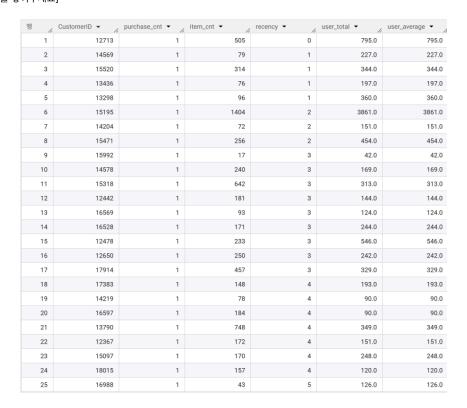
● 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

SELECT *
FROM enhanced-ward-466622-g5.1st_project.user_rfm;

[결과 이미지를 넣어주세요]



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 enhanced-ward-466622-g5.1st_project.user_data AS

WITH unique_products AS (

SELECT

CustomerID,

COUNT(DISTINCT StockCode) AS unique_products

FROM enhanced-ward-466622-g5.1st_project.Data1

GROUP BY CustomerID
)

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

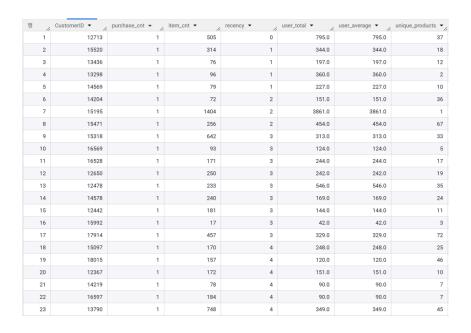
FROM enhanced-ward-466622-g5.1st_project.user_rfm AS ur

JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;

SELECT *

FROM enhanced-ward-466622-g5.1st_project.user_data
```

[결과 이미지를 넣어주세요]

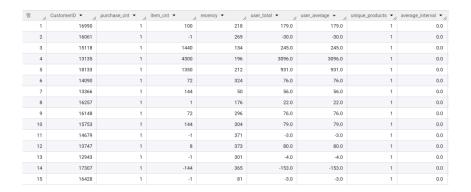


2. 평균 구매 주기

- 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 enhanced-ward-466622-g5.1st_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 interval_
   enhanced-ward-466622-g5.1st_project.Data1
  WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM enhanced-ward-466622-g5.1st_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]



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 enhanced-ward-466622-g5.1st_project.user_data AS
WITH TransactionInfo AS (
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS total_transactions,
  COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS cancel_frequency
 FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID
)
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(
   WHEN t.total_transactions > 0 THEN (t.cancel_frequency * 100.0 / t.total_transactions)
   ELSE 0
  END, 2
) AS cancel_rate
FROM enhanced-ward-466622-g5.1st_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

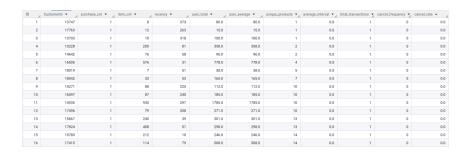
[결과 이미지를 넣어주세요]

● 이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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

```
# SELECT *
FROM enhanced-ward-466622-g5.1st_project.user_data;
```

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

Keep: SQL을 사용해 데이터를 특정 기준에 따라서 분류하고, 분류된 데이터를 제거하거나 연산을 통해 새로운 정보를 습득하는 능력을 유지할 것 Problem: 데이터 값의 특이점이나 null 값을 분석해서 잘못된 데이터를 분류했지만, 아직도 불량 데이터가 남아 있음.

Try : 이를 보완하기 위해서 데이터 수집이나 전처리 과정에서 먼저 불량 데이터를 거르고, 휴리스틱 분석을 통해 불량 데이터를 색출하는 방법도 공부해야 할 것이다.