고객을 세그먼테이션하자 [프로젝트] (홍성민) (1)

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT  *
FROM `curious-sandbox-466601-h2.modulabs_project.data` LIMIT 10
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT  COUNT(*)
FROM `curious-sandbox-466601-h2.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

행	f0_
1	541909

데이터 수 세기

- **COUNT 함수를 사용해서, 각 컬럼별 데이터 포인트의 수를 세어 보기**

```
SELECT COUNT(InvoiceNo),COUNT(StockCode),COUNT(Description),COUNT(Quantity),COUNT(InvoiceDate),COUNT(UnitPrice
FROM `curious-sandbox-466601-h2.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

행	f0_	f1_	f2_	f3_	f4_	f5_	f6_	f7_
1	541909	541909	540455	541909	541909	541909	406829	

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
   'Description' AS missing_data,
   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM curious-sandbox-466601-h2.modulabs_project.data

UNION ALL

SELECT
'CustomerID' AS missing_data,
   ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS  missing_percentage
   FROM curious-sandbox-466601-h2.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	missing_data	missing_percenta...
1	CustomerID	24.93
2	Description	0.27

결측치 처리 전략

- `StockCode = '85123A'` 의 `Description` 을 추출하는 쿼리문을 작성하기

```
SELECT DISTINCT StockCode,Description
FROM curious-sandbox-466601-h2.modulabs_project.data
WHERE StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

행	StockCode	Description
1	85123A	WHITE HANGING HEART T-LIG...
2	85123A	?
3	85123A	wrongly marked carton 22804
4	85123A	CREAM HANGING HEART T-LIG...

결측치 처리

- **DELETE 구문을 사용하며, WHERE 절을 통해 데이터를 제거할 조건을 제시**

```
DELETE
FROM curious-sandbox-466601-h2.modulabs_project.data
WHERE CustomerID IS NULL OR Description IS NULL
```

[결과 이미지를 넣어주세요]

이 문으로 data의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
WITH distinct_count AS(
        SELECT InvoiceNo,
        StockCode,
        Description,
        Quantity,
        InvoiceDate,
        UnitPrice,
        CustomerID,
        Country,
        COUNT(*) AS count_data
        FROM curious-sandbox-466601-h2.modulabs_project.data
        GROUP BY
        InvoiceNo,
        StockCode,
        Description,
        Quantity,
        InvoiceDate,
        UnitPrice,
        CustomerID,
        Country
        HAVING COUNT(*)>1   )
SELECT COUNT(*)
FROM distinct_count
```

[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 `curious-sandbox-466601-h2.modulabs_project.data` AS
SELECT DISTINCT *
FROM curious-sandbox-466601-h2.modulabs_project.data
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 테이블이 교체되었습니다.

남아있는데이터

행	f0_
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 `InvoiceNo`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM curious-sandbox-466601-h2.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	22190

- **고유한 `InvoiceNo`를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM curious-sandbox-466601-h2.modulabs_project.data
limit 100
```

[결과 이미지를 넣어주세요]

행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318

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

```
SELECT *
FROM curious-sandbox-466601-h2.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC
5	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC
6	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC
7	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC
8	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC
9	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC

- **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 curious-sandbox-466601-h2.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTiNCT StockCode)
FROM curious-sandbox-466601-h2.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	3684

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

```
SELECT StockCode,COUNT(*) AS sell_cnt
FROM curious-sandbox-466601-h2.modulabs_project.data
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

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM curious-sandbox-466601-h2.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	number_count	stock_cnt
1	5	3676
2	0	7
3	1	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)

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM curious-sandbox-466601-h2.modulabs_project.data
)
WHERE number_count = 0 or number_count = 1
```

[결과 이미지를 넣어주세요]

행	StockCode	number_count
1	POST	0
2	M	0
3	C2	1
4	D	0
5	BANK CHARGES	0
6	PADS	0
7	DOT	0
8	CRUK	0

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM curious-sandbox-466601-h2.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM curious-sandbox-466601-h2.modulabs_project.data
)
WHERE number_count = 0 or number_count = 1)
);
```

[결과 이미지를 넣어주세요]

이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS description_cnt
FROM curious-sandbox-466601-h2.modulabs_project.data
GROUP BY Description
```

[결과 이미지를 넣어주세요]

행	Description	description_cnt
1	MEDIUM CERAMIC TOP STORA...	208
2	BLACK CANDELABRA T-LIGHT ...	40
3	CAMOUFLAGE EAR MUFF HEA...	17
4	EMERGENCY FIRST AID TIN	126
5	BOX OF 6 ASSORTED COLOUR T...	75
6	ALARM CLOCK BAKELIKE PINK	636
7	FOUR HOOK WHITE LOVEBIRDS	265
8	RED DRAWER KNOB ACRYLIC E...	101
9	BLUE 3 PIECE POLKADOT CUTL...	97
10	COLOUR GLASS. STAR T-LIGHT ...	247

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM curious-sandbox-466601-h2.modulabs_project.data
WHERE
Description = 'Next Day Carriage' OR Description = 'High Resolution Image'
```

[결과 이미지를 넣어주세요]

이 문으로 data의 행 83개가 삭제되었습니다.

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE curious-sandbox-466601-h2.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  upper(Description) AS Description
FROM curious-sandbox-466601-h2.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT  MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM curious-sandbox-466601-h2.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

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

```
SELECT COUNT(UnitPrice) AS cnt_quantity, MIN(Quantity) AS min_quantity,MAX(Quantity) AS max_quantity, AVG(Quantity) AS
FROM curious-sandbox-466601-h2.modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

- **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]

이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate 컬럼을 연월일 자료형으로 변경하기**

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM curious-sandbox-466601-h2.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitP
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	
3	2010-12-07	537626	85116	12	2010-12-07 14:57:00 UTC	
4	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	
5	2010-12-07	537626	22729	4	2010-12-07 14:57:00 UTC	
6	2010-12-07	537626	21731	12	2010-12-07 14:57:00 UTC	
7	2010-12-07	537626	22375	4	2010-12-07 14:57:00 UTC	
8	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC	
9	2010-12-07	537626	22775	12	2010-12-07 14:57:00 UTC	

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX (InvoiceDate) AS most_recent_date

FROM curious-sandbox-466601-h2.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	most_recent_date
1	2011-12-09 12:50:00 UTC

- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

```
SELECT
   CustomerID,
   MAX (InvoiceDate) AS most_recent_date
FROM curious-sandbox-466601-h2.modulabs_project.data
GROUP BY CustomerID
```

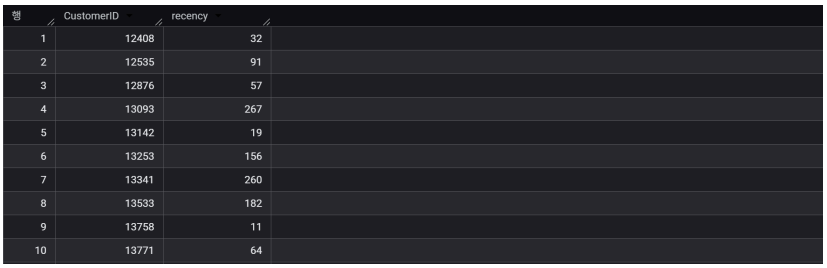
[결과 이미지를 넣어주세요]

행	CustomerID	most_recent_date
1	12346	2011-01-18 10:17:00 UTC
2	12347	2011-12-07 15:52:00 UTC
3	12348	2011-09-25 13:13:00 UTC
4	12349	2011-11-21 09:51:00 UTC
5	12350	2011-02-02 16:01:00 UTC
6	12352	2011-11-03 14:37:00 UTC
7	12353	2011-05-19 17:47:00 UTC
8	12354	2011-04-21 13:11:00 UTC
9	12355	2011-05-09 13:49:00 UTC
10	12356	2011-11-17 08:40:00 UTC

- **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]



행	CustomerID	recency
1	12408	32
2	12535	91
3	12876	57
4	13093	267
5	13142	19
6	13253	156
7	13341	260
8	13533	182
9	13758	11
10	13771	64

- **최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 `user_r` 이라는 이름의 테이블로 저장하기**

```
CREATE OR REPLACE TABLE curious-sandbox-466601-h2.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 curious-sandbox-466601-h2.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

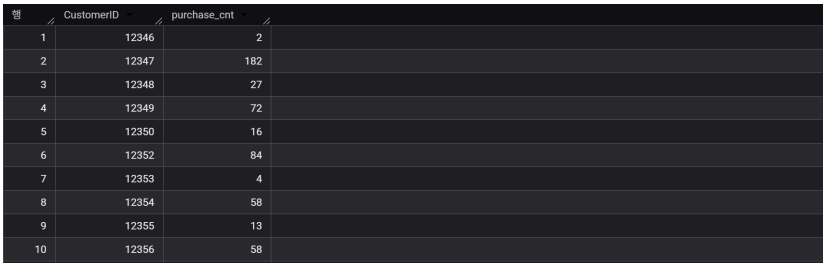
이 문으로 이름이 user_r인 테이블이 교체되었습니다.

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM curious-sandbox-466601-h2.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

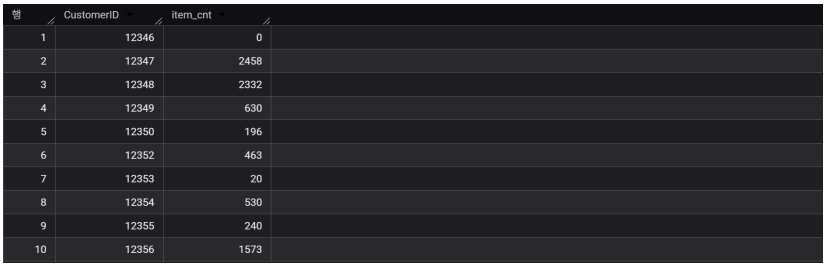


행	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84
7	12353	4
8	12354	58
9	12355	13
10	12356	58

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM curious-sandbox-466601-h2.modulabs_project.data
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

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

```
CREATE OR REPLACE TABLE curious-sandbox-466601-h2.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM curious-sandbox-466601-h2.modulabs_project.data
GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM curious-sandbox-466601-h2.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 curious-sandbox-466601-h2.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_rf인 테이블이 교체되었습니다.

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
 SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice),1) AS user_total
FROM curious-sandbox-466601-h2.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4

- **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 curious-sandbox-466601-h2.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 curious-sandbox-466601-h2.modulabs_project.user_rf AS rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice),1) AS user_total
FROM curious-sandbox-466601-h2.modulabs_project.data
GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_rfm인 테이블이 교체되었습니다.

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT*
FROM curious-sandbox-466601-h2.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	16446	4	2	0	2.9	1.0
2	14446	276	856	0	1005.6	4.0
3	15910	266	1013	0	1228.9	5.0
4	12748	4440	23516	0	29820.0	7.0
5	13069	469	5454	0	3713.1	8.0
6	17364	409	2671	0	4437.2	11.0
7	17315	482	3805	0	6153.3	13.0
8	15804	273	2513	0	3848.5	14.0
9	12423	118	1312	0	1624.1	14.0

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 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]

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 (
  -- (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_
    FROM
      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;
```

[결과 이미지를 넣어주세요]

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  curious-sandbox-466601-h2.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    -- # [[YOUR QUERY]] AS total_transactions,
    SUM(CASE WHEN InvoiceNo like 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
  FROM  curious-sandbox-466601-h2.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), (SELECT ROUND(
        SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/ COUNT(*)*100, 2)
FROM curious-sandbox-466601-h2.modulabs_project.data)AS cancel_rate

FROM curious-sandbox-466601-h2.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	cancel_frequency	cancel_rate
1	16138	1	-1	368	-8.0	-8.0	1	0.0	1	2.13
2	17925	1	72	372	244.1	244.0	1	0.0	0	2.13
3	17956	1	1	249	12.8	13.0	1	0.0	0	2.13
4	17763	1	12	263	15.0	15.0	1	0.0	0	2.13
5	18174	1	50	7	104.0	104.0	1	0.0	0	2.13
6	15389	1	400	172	500.0	500.0	1	0.0	0	2.13
7	17307	1	-144	365	-152.6	-153.0	1	0.0	1	2.13

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

```
SELECT*
FROM curious-sandbox-466601-h2.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15313	1	25	110	52.0	52.0	
2	12791	1	96	373	177.6	178.0	
3	13829	1	-12	359	-102.0	-102.0	
4	16737	1	288	53	417.6	418.0	
5	17986	1	10	56	20.8	21.0	
6	14679	1	-1	371	-2.5	-3.0	
7	13185	1	12	267	71.4	71.0	

회고

[회고 내용을 작성해주세요]

Keep :

Problem : 처음보는함수들의 조합에 값을 예상하기어렵고 진행도중의 값들이 잘되고있는지 파악하는것에문제가 있었다

Try :

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM curious-sandbox-466601-h2.modulabs_project.data;
```