고객을 세그먼테이션하자 [프로젝트] (1)

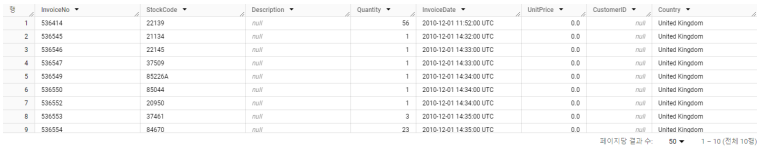
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM avid-involution-439402-i8.modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]



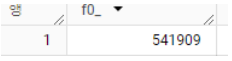
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom

페이지당 결과 수: 50 1 – 10 (전체 10행)

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM avid-involution-439402-i8.modulabs_project.data
```

[결과 이미지를 넣어주세요]



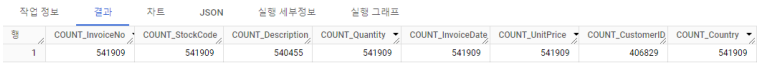
행	f0_
1	541909

데이터 수 세기

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

```
BigQuery에서는 각 컬럼별로 데이터 포인트의 수를 세는 것이 가능하지만, 모든 컬럼을 자동으로 한 번에 계산하는 기능은 없고,
```

[결과 이미지를 넣어주세요]



작업 정보 결과 자트 JSON 실행 세부정보 실행 그래프

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country
1	541909	541909	540455	541909	541909	541909	406829	541909

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 column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM evident-scion-439400-m6.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_percent
FROM evident-scion-439400-m6.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_perce
FROM evident-scion-439400-m6.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_percenta
FROM evident-scion-439400-m6.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_perce
FROM evident-scion-439400-m6.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_percent
FROM evident-scion-439400-m6.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_percen
FROM evident-scion-439400-m6.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_percentag
FROM evident-scion-439400-m6.modulabs_project.data ;
```

[결과 이미지를 넣어주세요]

column_name	missing_percentage
Quantity	0.0
Country	0.0
UnitPrice	0.0
CustomerID	24.93
InvoiceDate	0.0
StockCode	0.0
Description	0.27
InvoiceNo	0.0

결측치 처리 전략

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

```
SELECT Description
FROM avid-involution-439402-i8.modulabs_project.data
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

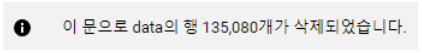
행	Description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	WHITE HANGING HEART T-LIG...

결측치 처리

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

```
DELETE FROM project_name.modulabs_project.data
WHERE CustomerID IS NULL OR Description IS NULL;
```

[결과 이미지를 넣어주세요]



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 project_name.modulabs_project.data
group by InvoiceNo,
StockCode,
Description,
Quantity,
InvoiceDate,
UnitPrice,
CustomerID,
Country
having count(*) > 1
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	557305	22645	CERAMIC HEART FAIRY CAKE ...	4	2011-06-19 14:42:00 UTC	1.45	13568	United Kingdom
2	569943	20972	PINK CREAM FELT CRAFT TRI...	1	2011-10-06 18:08:00 UTC	1.25	14592	United Kingdom
3	571241	72816	SET/3 CHRISTMAS DECOUPAG...	1	2011-10-14 14:58:00 UTC	0.95	14592	United Kingdom
4	571241	22095	LADS ONLY TISSUE BOX	3	2011-10-14 14:58:00 UTC	0.39	14592	United Kingdom
5	571241	22630	DOLLY GIRL LUNCH BOX	1	2011-10-14 14:58:00 UTC	1.95	14592	United Kingdom
6	571241	22940	FELTCRAFT CHRISTMAS FAIRY	1	2011-10-14 14:58:00 UTC	4.25	14592	United Kingdom
7	571241	22807	SET OF 6 T-LIGHTS TOADSTOO...	1	2011-10-14 14:58:00 UTC	2.95	14592	United Kingdom
8	554917	22848	BREAD BIN DINER STYLE PINK	1	2011-05-27 12:29:00 UTC	16.95	15104	United Kingdom
9	554917	22849	BREAD BIN DINER STYLE MINT	1	2011-05-27 12:29:00 UTC	16.95	15104	United Kingdom

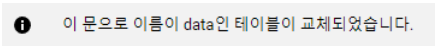
페이지당 결과 수: 50 1 – 50 (전체 4837행)

중복값 처리

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

```
CREATE OR REPLACE TABLE evident-scion-439400-m6.modulabs_project.data AS
SELECT DISTINCT *
FROM evident-scion-439400-m6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

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

```
select count(distinct InvoiceNo)
from evident-scion-439400-m6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

f0_
22190

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

```
select distinct InvoiceNo
from evident-scion-439400-m6.modulabs_project.data
limit 100
```

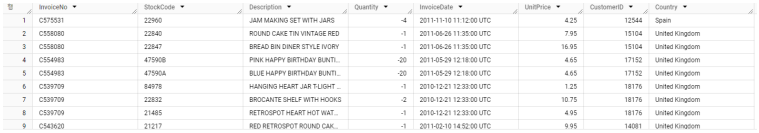
[결과 이미지를 넣어주세요]

행	InvoiceNo
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032
7	561387
8	574868
9	574827

- **`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;
```

[결과 이미지를 넣어주세요]



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
2	C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
3	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
4	C554983	47590B	PINK HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
6	C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom
7	C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
8	C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom
9	C543620	21217	RED RETROSPOT ROUND CAK...	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom

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

```
SELECT ROUND(SUM(CASE WHEN # [[YOUR QUERY]] THEN 1 ELSE 0 END)/ COUNT(*) * 100, 1)
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

f0_
2.2

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
select count(distinct StockCode)
from evident-scion-439400-m6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

f0_
3684

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM evident-scion-439400-m6.modulabs_project.data
group by StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

StockCode	sell_cnt
85123A	2065
22423	1894
85099B	1659
47566	1409
84879	1405
20725	1346
22720	1224
POST	1196
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 project_name.modulabs_project.data
)
WHERE number_count in (0,1);
```

[결과 이미지를 넣어주세요]

StockCode	number_count
POST	0
M	0
PADS	0
D	0
BANK CHARGES	0
DOT	0
CRUK	0
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)

```
SELECT ROUND(SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0
FROM evident-scion-439400-m6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

f0_
0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM evident-scion-439400-m6.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 evident-scion-439400-m6.modulabs_project.data
)
WHERE number_count in (0,1)
)
```

[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 project_name.modulabs_project.data
group by Description
ORDER BY description_cnt desc
LIMIT 30
```

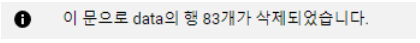
[결과 이미지를 넣어주세요]

Description	description_cnt
WHITE HANGING HEART T-LIG...	2058
REGENCY CAKESTAND 3 TIER	1894
JUMBO BAG RED RETROSPOT	1659
PARTY BUNTING	1409
ASSORTED COLOUR BIRD ORN...	1405
LUNCH BAG RED RETROSPOT	1345
SET OF 3 CAKE TINS PANTRY ...	1224
LUNCH BAG BLACK SKULL.	1099
PACK OF 72 RETROSPOT CAKE...	1062

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM project_name.modulabs_project.data
WHERE
WHERE Description = 'Next Day Carriage' or Description = 'High Resolution Image'
```

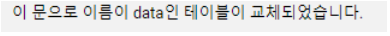
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

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

min_price	max_price	avg_price
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_quantit
FROM evident-scion-439400-m6.modulabs_project.data
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

cnt_quantity	min_quantity	max_quantity	avg_quantity
33	1	12540	420.5151515151...

- `UnitPrice = 0` **를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE evident-scion-439400-m6.modulabs_project.data AS
SELECT *
FROM evident-scion-439400-m6.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 evident-scion-439400-m6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	InvoiceDay
1	2011-11-03
2	2011-11-03
3	2011-11-03
4	2011-11-03
5	2011-11-03
6	2011-11-03
7	2011-11-03
8	2011-11-03
9	2011-11-03

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
FROM evident-scion-439400-m6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

most_recent_date
2011-12-09

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

```
SELECT
    CustomerID,
```

```
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM evident-scion-439400-m6.modulabs_project.data
group by CustomerID;
```

[결과 이미지를 넣어주세요]

CustomerID	InvoiceDay
12544	2011-11-10
13568	2011-06-19
13824	2011-11-07
14080	2011-11-07
14336	2011-11-23
14592	2011-11-04
15104	2011-06-26
15360	2011-10-31
15872	2011-11-25

- **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]

CustomerID	recency
14342	21
14856	36
12559	310
13583	155
17690	30
12842	70
17728	3
17761	32
12401	303

- **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 evident-scion-439400-m6.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

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]

CustomerID	purchase_cnt
12544	19
13568	43
13824	46
14080	4
14336	90
14592	158
15104	69
15360	13
15872	108

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

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

[결과 이미지를 넣어주세요]

CustomerID	item_cnt
12544	130
13568	66
13824	768
14080	48
14336	1759
14592	407
15104	633
15360	223
15872	187

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

```
CREATE OR REPLACE TABLE evident-scion-439400-m6.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
  CustomerID,
  Count(InvoiceNo) AS purchase_cnt
FROM evident-scion-439400-m6.modulabs_project.data
```

```
group by CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
  CustomerID,
  sum(Quantity) AS item_cnt
FROM evident-scion-439400-m6.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 evident-scion-439400-m6.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

```
이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.
```

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  sum(UnitPrice) AS user_total
FROM project_name.modulabs_project.data
group by CustomerID;
```

[결과 이미지를 넣어주세요]

CustomerID	user_total
12544	54.0
13568	131.0
13824	127.0
14080	4.0
14336	145.0
14592	324.0
15104	365.0
15360	31.0
15872	226.0

- **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 evident-scion-439400-m6.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 evident-scion-439400-m6.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
  ROUND(sum(UnitPrice)) AS user_total
FROM evident-scion-439400-m6.modulabs_project.data
group by CustomerID
) 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 evident-scion-439400-m6.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
16558	474	5346	0	1441.0	3.040084388185...
14422	222	2906	0	624.0	2.810810810810...
16705	284	5458	0	1284.0	4.521126760563...
13426	158	2225	0	300.0	1.898734177215...
12662	222	2023	0	561.0	2.527027027027...
17490	85	1022	0	246.0	2.894117647058...
17001	169	2164	0	517.0	3.059171597633...
14446	276	856	0	510.0	1.847826086956...
12433	420	11071	0	921.0	2.192857142857...

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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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 evident-scion-439400-m6.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
  count(*)   AS total_transactions,
  count(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE NULL END) AS cancel_frequency
  FROM evident-scion-439400-m6.modulabs_project.data
```

```
  group by CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID),ROUND(cancel_frequency / total_transactions * 100, 2) AS cancel_rate
FROM `evident-scion-439400-m6.modulabs_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 evident-scion-439400-m6.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
16323	1	50	196	4.0	4.0	1	0.0	1	0	0.0
15657	1	24	22	1.0	1.0	1	0.0	1	0	0.0
13270	1	200	366	3.0	3.0	1	0.0	1	0	0.0
16738	1	3	297	1.0	1.0	1	0.0	1	0	0.0
16454	1	2	64	3.0	3.0	1	0.0	1	0	0.0
14351	1	12	164	4.0	4.0	1	0.0	1	0	0.0
16148	1	72	296	1.0	1.0	1	0.0	1	0	0.0
17443	1	504	219	1.0	1.0	1	0.0	1	0	0.0
17986	1	10	56	2.0	2.0	1	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]