

고객을 세그먼테이션하자 [프로젝트] (1)

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM dauntless-karma-466601-p1.modulabs_project.data  
LIMIT 10
```

행	InvoiceNo	StockCode	Description	Quantity	Invoice
1	541431	23166	MEDIUM CERAMIC TOP STORA...	74215	2011-0
2	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-0
3	537626	22494	EMERGENCY FIRST AID TIN	12	2010-1
4	537626	21171	BATHROOM METAL SIGN	12	2010-1
5	537626	22771	CLEAR DRAWER KNOB ACRYLI...	12	2010-1
6	537626	22726	ALARM CLOCK BAKELIKE GREEN	4	2010-1
7	537626	84997C	BLUE 3 PIECE POLKADOT CUTL...	6	2010-1
8	537626	22775	PURPLE DRAWERKNOB ACRYLI...	12	2010-1

페이지당 결과 수: 50 1 - 10 (전체 10행) |< < > >|

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)  
FROM dauntless-karma-466601-p1.modulabs_project.data
```

행	f0_
1	406829

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 -- 540455
, COUNT(Quantity) AS COUNT_Quantity
, COUNT(InvoiceDate) AS COUNT_InvoiceDate
, COUNT(UnitPrice) AS COUNT_UnitPrice
, COUNT(CustomerID) AS COUNT_CustomerID -- 406829
, COUNT(Country) AS COUNT_Country
FROM dauntless-karma-466601-p1.modulabs_project.data
```

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	COUNT_UnitPrice	COUN
1	406829	406829	406829	406829	406829	406829	

페이지당 결과 수: 50

1 - 1 (전체 1행)

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(*)) AS missing_percentage
FROM dauntless-karma-466601-p1.modulabs_project.data
UNION ALL
SELECT
  'Description' AS column_name,
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*)) AS missing_percentage
FROM dauntless-karma-466601-p1.modulabs_project.data
UNION ALL
SELECT
  'CustomerID' AS column_name,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*)) AS missing_percentage
FROM dauntless-karma-466601-p1.modulabs_project.data
```

135,080 행을 삭제한 후에 다시 캡처를 떠서 결과가 바뀌었습니다ㅠㅠ 양해바랍니다

작업 정보 <u>결과</u> 차트 JSON 실행 세부정보 실행 그래프			
행	column_name ▼	missing_percenta...	
1	Description	0.0	
2	CustomerID	0.0	
3	InvoiceNo	0.0	

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE StockCode = '85123A'
```

135,080 행을 삭제한 후에 다시 캡처를 떠서 결과가 줄었습니다

작업 정보			결과	차트	JSON	실행 세부정보	실행 그래프
행	Description ▼						
1	WHITE HANGING HEART T-LIG...						
2	CREAM HANGING HEART T-LIG...						

결측치 처리

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

```
DELETE FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE Description IS NULL OR CustomerID IS NULL
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 135,080개가 삭제되었습니다.

테이블로 이동

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 InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPri
, COUNT(*) AS DUP
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, Unit
HAVING COUNT(*) > 1
```

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	Invoice
43	575519	23506	MINI PLAYING CARDS SPACEB...	20	2011-1
44	542106	20676	RED RETROSPOT BOWL	24	2011-C
45	564539	22378	WALL TIDY RETROSPOT	5	2011-C
46	561658	21108	FAIRY CAKE FLANNEL ASSORT...	1	2011-C
47	561661	21591	COSY HOUR CIGAR BOX MATC...	1	2011-C
48	561661	22289	HANGING METAL CHICKEN DE...	2	2011-C
49	537136	22694	WICKER STAR	1	2010-1
50	537136	85049A	TRADITIONAL CHRISTMAS RIB...	2	2010-1

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 dauntless-karma-466601-p1.modulabs_pro
AS SELECT DISTINCT * FROM dauntless-karma-466601-p1.modulabs_pro
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

테이블로 이동

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
```

```
FROM dauntless-karma-466601-p1.modulabs_project.data
```

행	f0_ ▼
1	22190

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

```
SELECT DISTINCT InvoiceNo
FROM dauntless-karma-466601-p1.modulabs_project.data
LIMIT 100
```

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

페이지당 결과 수: 50 ▼ 1 - 50 (전체 100행) |< < > >|

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

```
SELECT *
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100
```

작업 탐색기					
결과					
차트					
JSON					
실행 세부정보					
실행 그래프					
행	InvoiceNo	StockCode	Description	Quantity	Invoice
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-C
2	C545329	M	Manual	-1	2011-C
3	C545329	M	Manual	-1	2011-C
4	C545330	M	Manual	-1	2011-C
5	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-C
6	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-C
7	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-C
8	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-C

페이지당 결과 수: 50 1 - 50 (전체 100행) < > >>

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

```
SELECT
  ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / C
FROM dauntless-karma-466601-p1.modulabs_project.data
```

행	Cancelled_Ratio
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM dauntless-karma-466601-p1.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 dauntless-karma-466601-p1.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

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

페이지당 결과 수: 50 ▼ 1 - 10 (전체 10행) |< < > >|

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

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
```



```

LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]',
FROM dauntless-karma-466601-p1.modulabs_project.data
)
WHERE number_count BETWEEN 0 AND 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

- **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(SUM(CASE WHEN StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CH
/ COUNT(StockCode) * 100, 2) AS code_ratio
FROM dauntless-karma-466601-p1.modulabs_project.data

```

행	code_ratio ▼
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE
FROM dauntless-karma-466601-p1.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 dauntless-karma-466601-p1.modulabs_project.data
    )
    WHERE number_count BETWEEN 0 AND 1
  )
)
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 1,915개가 삭제되었습니다.

테이블로 이동

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30
```

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

• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE Description IN('Next Day Carriage', 'Next Day Carriage')
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 80개가 삭제되었습니다.

테이블로 이동

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE dauntless-karma-466601-p1.modulabs_proje
SELECT
* EXCEPT (Description),
```

```
UPPER(Description) AS Description
FROM dauntless-karma-466601-p1.modulabs_project.data
```

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

테이블로 이동

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 dauntless-karma-466601-p1.modulabs_project.data
```

행	min_price ▼	max_price ▼	avg_price ▼
1	0.0	649.5	2.904998761289...

- 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 avg_quantity
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE UnitPrice = 0
```

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 dauntless-karma-466601-p1.modulabs_project
SELECT *
FROM dauntless-karma-466601-p1.modulabs_project.data
WHERE UnitPrice <> 0
```

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

테이블로 이동

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM dauntless-karma-466601-p1.modulabs_project.data
```

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-04-14	550188	22636	1	2011-04-14 18:57:00
2	2010-12-05	537197	22841	1	2010-12-05 14:02:00
3	2011-11-07	574920	22899	1	2011-11-07 16:34:00
4	2011-11-07	574920	23480	1	2011-11-07 16:34:00
5	2011-01-13	541109	22168	1	2011-01-13 15:10:00
6	2011-07-26	561284	22167	1	2011-07-26 12:24:00
7	2011-11-18	577314	23407	2	2011-11-18 13:23:00
8	2011-11-07	574879	22625	2	2011-11-07 13:22:00

페이지당 결과 수: 50 1 - 33 (전체 33행) < >

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT DATE(MAX(InvoiceDate)) AS most_recent_date
FROM dauntless-karma-466601-p1.modulabs_project.data
```

행	most_recent_date
1	2011-12-09

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM dauntless-karma-466601-p1.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 recenc
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	15602	8
2	15107	316
3	14410	235
4	14646	91
5	16560	344
6	15804	50
7	12457	225
8	14110	22
9	12444	7

페이지당 결과 수: 50 1 - 24 (전체 24행) |< < > >|

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

```
CREATE OR REPLACE TABLE dauntless-karma-466601-p1.modulabs_project.user_r
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(InvoiceDay) AS InvoiceDay
  FROM dauntless-karma-466601-p1.modulabs_project.data
  GROUP BY CustomerID
);
```

i 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

테이블로 이동

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
```

```

COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY CustomerID

```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
1	12457	1
2	12647	1
3	13985	1
4	15107	1
5	16818	1
6	12444	1
7	13014	1
8	14410	1
9	15804	1

페이지당 결과 수: 50 1 - 24 (전체 24행) |< < > >|

• 각 고객 별로 구매한 아이템의 총 수량 더하기

```

SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY CustomerID

```

행	CustomerID	item_cnt
1	12457	1
2	12647	1
3	13985	2
4	15107	1
5	16818	1
6	12444	2
7	13014	2
8	14410	2
9	15804	2

페이지당 결과 수: 50 1 - 24 (전체 24행) |< < > >|

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


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

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  # [[YOUR QUERY]]
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  # [[YOUR QUERY]]
)

-- 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;
```

[결과 이미지를 넣어주세요]

 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

테이블로 이동

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY CustomerID

```

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.6
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행) |< < > >|

• 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 dauntless-karma-466601-p1.modulabs_proje
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 dauntless-karma-466601-p1.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity), 1) AS user_total

```

```
FROM dauntless-karma-466601-p1.modulabs_project.data
GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID
ORDER BY rf.purchase_cnt DESC
```

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

테이블로 이동

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT * FROM dauntless-karma-466601-p1.modulabs_project.user_rfm
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	14911	242	76823	1	128768.2	532.1
2	12748	217	23516	0	29820.0	137.4193548387...
3	17841	169	22613	1	39861.5	235.8668639053...
4	14606	125	5932	1	11486.6	91.892800000000...
5	15311	118	37673	0	59284.2	502.4084745762...
6	13089	118	30742	2	57322.1	485.7805084745...
7	12971	88	9204	3	10933.8	124.2477272727...
8	13408	75	16128	1	27888.4	371.8453333333...

페이지당 결과 수: 50 1 - 50 (전체 4362행) < >

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;
```

i 이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

테이블로 이동

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
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
```

```

        CustomerID,
        DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID
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;

```

i 이 문으로 이름이 user_data인 테이블이 교체되었습니다.

테이블로 이동

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 dauntless-karma-466601-p1.modulabs_project.user_data
WITH TransactionInfo AS (
    SELECT CustomerID,
        COUNT(InvoiceNo) AS total_transactions,
        SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
    FROM dauntless-karma-466601-p1.modulabs_project.data
    GROUP BY CustomerID
)

```

```
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_trar
FROM `dauntless-karma-466601-p1.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.

테이블로 이동

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

```
SELECT * FROM dauntless-karma-466601-p1.modulabs_project.user_data
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	un
1	12814	1	48	101	85.9	85.9	
2	13120	1	12	238	30.6	30.6	
3	14576	1	12	372	35.4	35.4	
4	14090	1	72	324	76.3	76.3	
5	12943	1	-1	301	-3.8	-3.8	
6	16881	1	600	66	432.0	432.0	
7	17443	1	504	219	534.2	534.2	
8	15524	1	4	24	440.0	440.0	

회고

[회고 내용을 작성해주세요]

Keep : 하루동안 특정 지점에서 오래 멈추지않고 마무리로 진전을 이루었다는것

Problem : 문제풀이, 코드복사, 결과화면 복사에 집중하다보니 정작 중요한 결과물의 검증
을 매 단계별로 꼼꼼히 하지 못했다. 그래서 데이터를 3번이나 다시 밀어넣으며 노가다로 검
증을 했다.(지금도 결과가 100% 맞게 되었는지는 다 확인하지 못한 상황)

Try : 좀더 여유를 갖고 처음부터 잘 검증한다