

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM seismic-pursuit-456102-a5.modulabs_project.data  
LIMIT 10  
# 8개의 컬럼
```

명	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	851234	WHITE HANGING HEART T-LIG...	6	2010-12-01 09:26:00 UTC	2.35	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 09:26:00 UTC	3.39	17850	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)  
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

f0_ ▼
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),  
COUNT(CustomerID),  
COUNT(Country)  
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

명	f0_	f1_	f2_	f3_	f4_	f5_	f6_	f7_
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_percentage  
FROM `seismic-pursuit-456102-a5.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 `seismic-pursuit-456102-a5.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_percentage
FROM `seismic-pursuit-456102-a5.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 `seismic-pursuit-456102-a5.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_percentage
FROM `seismic-pursuit-456102-a5.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 `seismic-pursuit-456102-a5.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 `seismic-pursuit-456102-a5.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 `seismic-pursuit-456102-a5.modulabs_project.data`;

#

```

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

결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM seismic-pursuit-456102-a5.modulabs_project.data
WHERE StockCode = '85123A'

```

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

결측치 처리

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

```
DELETE FROM `seismic-pursuit-456102-a5.modulabs_project.data`
WHERE CustomerID IS NULL
OR Description IS NULL;
```

이 문으로 data1의 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 COUNT(*) AS duplicate_count
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS cnt
  FROM `seismic-pursuit-456102-a5.modulabs_project.data`
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  HAVING cnt > 1
);
```

행	duplicate_count
1	4837

중복값 처리

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

```
CREATE OR REPLACE TABLE `seismic-pursuit-456102-a5.modulabs_project.data` AS
SELECT DISTINCT *
FROM `seismic-pursuit-456102-a5.modulabs_project.data`;
```

```
# 남은 데이터 행 개수 확인
SELECT COUNT(*)
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

이 문으로 이름이 data1인 테이블이 교체되었습니다.

테이블로 이동

행	f0_
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

f0_
22190

- 고유한 InvoiceNo를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM seismic-pursuit-456102-a5.modulabs_project.data
LIMIT 100
```

행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201

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

```
SELECT *
FROM seismic-pursuit-456102-a5.modulabs_project.data
WHERE InvoiceNo LIKE "C%"
LIMIT 100;
```

2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352
5	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352
6	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352
7	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12	2011-03-22 16:07:00 UTC	1.49	12352
8	C547388	22545	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352
9	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03-22 16:07:00 UTC	2.95	12352

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

```
SELECT
  ROUND(
    COUNTIF(STARTS_WITH(InvoiceNo, 'C')) / COUNT(InvoiceNo) * 100,
    1
  ) AS canceled_percentage
FROM `seismic-pursuit-456102-a5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	canceled_percentage
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

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 seismic-pursuit-456102-a5.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

- **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]', '')) AS number_count
  FROM project_name.modulabs_project.data
)
WHERE number_count = 0 OR number_count = 1

```

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(
    (COUNTIF(number_count = 0) + COUNTIF(number_count = 1)) / COUNT(*) * 100, 2
  ) AS stockcode_0_1_percentage
FROM (
  SELECT
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `seismic-pursuit-456102-a5.modulabs_project.data`
)

```

stockcode_0_1_perce
0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```

DELETE FROM seismic-pursuit-456102-a5.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 seismic-pursuit-456102-a5.modulabs_project.data
  )
  WHERE number_count = 0 OR number_count = 1
)

```

i 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 seismic-pursuit-456102-a5.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 ...	1224

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM seismic-pursuit-456102-a5.modulabs_project.data
WHERE Description = 'Next Day Carriage' OR Description = 'High Resolution Image'
```

i 이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE seismic-pursuit-456102-a5.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

▼ 확인해보자

```
SELECT DISTINCT Description
FROM seismic-pursuit-456102-a5.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

i 표시할 데이터가 없습니다.

UnitPrice 살펴보기

- UnitPrice 의 최소값, 최대값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757405...

- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최소값, 최대값, 평균 구하기

```
SELECT COUNT(UnitPrice) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS
FROM seismic-pursuit-456102-a5.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 seismic-pursuit-456102-a5.modulabs_project.data AS
SELECT *
FROM seismic-pursuit-456102-a5.modulabs_project.data
WHERE UnitPrice != 0
```

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay
FROM seismic-pursuit-456102-a5.modulabs_project.data
```

행	InvoiceDay
1	2011-01-18
2	2011-01-18
3	2010-12-07
4	2010-12-07
5	2010-12-07
6	2010-12-07
7	2010-12-07

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  (SELECT MAX(InvoiceDate)) FROM seismic-pursuit-456102-a5.modulabs_project.data) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM seismic-pursuit-456102-a5.modulabs_project.data;
```

행	most_recent_date	InvoiceDay	InvoiceNo
1	2011-12-09	2011-01-18	541431
2	2011-12-09	2011-01-18	C541433
3	2011-12-09	2010-12-07	537626
4	2011-12-09	2010-12-07	537626
5	2011-12-09	2010-12-07	537626
6	2011-12-09	2010-12-07	537626

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(InvoiceDate) AS recent_purchase_date
FROM seismic-pursuit-456102-a5.modulabs_project.data
GROUP BY CustomerID
```

행	CustomerID	recent_purchase_date
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19

- 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(InvoiceDate) AS InvoiceDay
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

행	CustomerID	recency
1	12475	53
2	12678	42
3	13117	21
4	13192	95
5	13656	164
6	13769	2
7	13974	49

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

```
CREATE OR REPLACE TABLE seismic-pursuit-456102-a5.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 seismic-pursuit-456102-a5.modulabs_project.data
  GROUP BY CustomerID
);

# 확인
# SELECT *
# FROM seismic-pursuit-456102-a5.modulabs_project.user_r
```

행	CustomerID	recency
1	13113	0
2	12526	0
3	12662	0
4	17490	0
5	15804	0
6	13426	0
7	15344	0

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `seismic-pursuit-456102-a5.modulabs_project.data`
GROUP BY CustomerID;
```

행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM seismic-pursuit-456102-a5.modulabs_project.data
GROUP BY CustomerID
```

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

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

```
CREATE OR REPLACE TABLE seismic-pursuit-456102-a5.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM seismic-pursuit-456102-a5.modulabs_project.data
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM seismic-pursuit-456102-a5.modulabs_project.data
  GROUP BY CustomerID
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.recentry
```

```

FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN seismic-pursuit-456102-a5.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;

# 확인용
# SELECT *
# FROM seismic-pursuit-456102-a5.modulabs_project.user_rf

```

i 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13436	1	76	1
3	14569	1	79	1
4	15520	1	314	1
5	13298	1	96	1
6	15471	1	256	2
7	14204	1	72	2

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
FROM `seismic-pursuit-456102-a5.modulabs_project.data`
GROUP BY CustomerID;

```

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0

- 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

```

SELECT
  rf.CustomerID,
  ROUND(ut.user_total / rf.purchase_cnt, 1) AS user_average
FROM seismic-pursuit-456102-a5.modulabs_project.user_rf rf
LEFT JOIN (
  SELECT
    CustomerID,

```

```

ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
FROM seismic-pursuit-456102-a5.modulabs_project.data
GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;

```

행	CustomerID	user_average
1	12713	794.5
2	13436	196.9
3	14569	227.4
4	15520	343.5
5	13298	360.0
6	15471	454.5
7	14204	150.6

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```

CREATE OR REPLACE TABLE seismic-pursuit-456102-a5.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, 1) AS user_average
FROM seismic-pursuit-456102-a5.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액 계산
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
  FROM seismic-pursuit-456102-a5.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;

# SELECT *
# FROM seismic-pursuit-456102-a5.modulabs_project.user_rfm

```

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

테이블로 이동

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	794.5
2	13436	1	76	1	196.9	196.9
3	15520	1	314	1	343.5	343.5
4	13298	1	96	1	360.0	360.0
5	14569	1	79	1	227.4	227.4
6	14204	1	72	2	150.6	150.6
7	15195	1	1404	2	3861.0	3861.0

▼ 고유한 유저의 수

```

SELECT COUNT(DISTINCT CustomerID)
FROM seismic-pursuit-456102-a5.modulabs_project.user_rfm

```

fo_ 4362

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;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	16257	1	1	176	21.9	21.9	1
2	13270	1	200	366	590.0	590.0	1
3	16765	1	4	294	34.0	34.0	1
4	17307	1	-144	365	-152.6	-152.6	1
5	13017	1	48	7	204.0	204.0	1
6	14705	1	100	198	179.0	179.0	1
7	18133	1	1350	212	931.5	931.5	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
  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;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	16257	1	1	176	21.9	21.9	1	0.0
2	13270	1	200	366	590.0	590.0	1	0.0
3	16765	1	4	294	34.0	34.0	1	0.0
4	17307	1	-144	365	-152.6	-152.6	1	0.0
5	13017	1	48	7	204.0	204.0	1	0.0
6	14705	1	100	198	179.0	179.0	1	0.0
7	18133	1	1350	212	931.5	931.5	1	0.0

3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기
 - 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
 - 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 `seismic-pursuit-456102-a5.modulabs_project.user_data` AS
```

```
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT IF(STARTS_WITH(InvoiceNo, 'C'), InvoiceNo, NULL)) AS cancel_frequency
  FROM `seismic-pursuit-456102-a5.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
```

```
SELECT
  u.*
  t.* EXCEPT(CustomerID),
  ROUND(t.cancel_frequency / t.total_transactions * 100, 2) AS cancel_rate
FROM `seismic-pursuit-456102-a5.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13403	1	44	17	75.0	75.0	3	0.0	1	0	0.0
2	16287	1	298	169	142.5	142.5	8	0.0	1	0	0.0
3	13751	1	152	288	296.3	296.3	9	0.0	1	0	0.0
4	12482	1	92	323	198.8	198.8	10	0.0	1	0	0.0
5	14208	1	114	94	151.7	151.7	10	0.0	1	0	0.0
6	12902	1	66	264	138.7	138.7	10	0.0	1	0	0.0
7	17690	1	96	23	184.4	184.4	11	0.0	1	0	0.0

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

```
SELECT *
FROM seismic-pursuit-456102-a5.modulabs_project.user_data
```

회고

Keep

1. `REPLACE` 구문 사용 시 테이블이 실제로 교체되었는지 `SELECT`로 바로 확인한 점
→ 덕분에 이후에도 오류 없이 원활하게 진행할 수 있었음

Problem	<p>1. 초반에 노선에 기록해야 할 실행 결과 사진을 저장하지 않았음 → 새로운 데이터 테이블 만들어서 다시 실행하고, 원래 데이터 테이블에 덮어 씌워서 해결. (빠른 대처가 좋았다고 생각함)</p> <p>2. 중간 DELETE 함수를 쓰다가 잘못 지우는 상황 발생. 복구가 불가능함. → 1번을 위해 만든 새 데이터 테이블을 통해 해결 → 백업 테이블 만드는 법을 배워서 활용하기로 함 → 삭제하기 전 <code>begin transaction;</code> 을 이용해 실행 확인!</p>
try	<p>1. 시작하기 전 보고서 형태를 본다면지 준비 확실하게 하고 프로젝트 시작하기.</p> <p>2. <code>DELETE</code> , <code>REPLACE</code> 등의 데이터를 변경해야하는 구문을 사용할 때는 백업 테이블을 만들고, 실행 확인하고 진행하는 습관 만들기</p> <p>3. 인사이트 해석을 하는데 있어 아직 미숙한 부분이 있음. 따로 더 공부해야 될 듯.</p>