

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM my-first-project-466601.modulabs_project.data  
LIMIT 10
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84404B	CREAM CUPID HEARTS COATH...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)  
FROM my-first-project-466601.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) 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 count_UnitPrice,  
COUNT(CustomerID) AS count_CustomerID,  
COUNT(Country) AS count_Country  
FROM my-first-project-466601.modulabs_project.data]
```

[결과 이미지를 넣어주세요]

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 ALL**을 통해 합치기

```
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(IF(InvoiceNo IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data

UNION ALL

SELECT
    'StockCode' AS column_name,
    ROUND(SUM(IF(StockCode IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data

UNION ALL

SELECT
    'Description' AS column_name,
    ROUND(SUM(IF(Description IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data

UNION ALL

SELECT
    'Quantity' AS column_name,
    ROUND(SUM(IF(Quantity IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data

UNION ALL

SELECT
    'InvoiceDate' AS column_name,
    ROUND(SUM(IF(InvoiceDate IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data

UNION ALL

SELECT
    'UnitPrice' AS column_name,
    ROUND(SUM(IF(UnitPrice IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data

UNION ALL

SELECT
    'CustomerID' AS column_name,
    ROUND(SUM(IF(CustomerID IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data

UNION ALL

SELECT
    'Country' AS column_name,
    ROUND(SUM(IF(Country IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
```

[결과 이미지를 넣어주세요]

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

결측치 처리 전략

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

```
SELECT DISTINCT(Description)
FROM my-first-project-466601.modulabs_project.data
WHERE StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

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 my-first-project-466601.modulabs_project.data
WHERE Description IS NULL
OR CustomerID 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

```
SELECT COUNT(*)
FROM (
  SELECT *
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
)
```

[결과 이미지를 넣어주세요]

행	f0_ ▼
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 my-first-project-466601.modulabs_project.data AS
SELECT DISTINCT *
FROM my-first-project-466601.modulabs_project.data;

SELECT COUNT(*)
FROM my-first-project-466601.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

 이 문으로 이름이 data인 테이블이 교체되었습니다.

쿼리 결과

작업 정보		결과	차트
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 my-first-project-466601.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	22190

- 고유한 **InvoiceNo**를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM my-first-project-466601.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 my-first-project-466601.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	CS41423	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12846	United Kingdom
2	CS45329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12852	Norway
3	CS45329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12852	Norway
4	CS46350	M	Manual	-1	2011-03-01 15:49:00 UTC	378.5	12852	Norway
5	CS47388	23545	CERAMIC HEART FAIRY CAKE...	-12	2011-03-22 16:07:00 UTC	1.48	12852	Norway
6	CS47388	37448	CERAMIC CAFE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12852	Norway
7	CS47388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	12852	Norway
8	CS47388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65	12852	Norway
9	CS47388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12852	Norway
10	CS47388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12852	Norway

- 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)
FROM my-first-project-466601.modulabs_project.data
```

```
SELECT ROUND(SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) * 100 / COUNT(*), 1)
FROM my-first-project-466601.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM my-first-project-466601.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 my-first-project-466601.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 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 my-first-project-466601.modulabs_project.data
)
WHERE number_count IN (0, 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)

```
WITH UNUSUAL_CODE AS(
  SELECT DISTINCT StockCode, number_count
  FROM (
    SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM my-first-project-466601.modulabs_project.data
  )
  WHERE number_count IN (0, 1)
)
SELECT
  ROUND(COUNT(CASE WHEN StockCode IN (SELECT StockCode FROM UNUSUAL_CODE) THEN 1 END) * 100 / COUNT(*), 2)
FROM my-first-project-466601.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM my-first-project-466601.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 my-first-project-466601.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 my-first-project-466601.modulabs_project.data  
GROUP BY Description  
ORDER BY description_cnt DESC  
LIMIT 30
```

[결과 이미지를 넣어주세요]

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
10	SPOTTY BUNTING	1026

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE  
FROM my-first-project-466601.modulabs_project.data  
WHERE  
UPPER(Description) IN (UPPER('Next Day Carriage'), UPPER('High Resolution Image'))
```

[결과 이미지를 넣어주세요]

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE my-first-project-466601.modulabs_project.data AS  
SELECT  
* EXCEPT (Description),  
UPPER(Description) AS Description  
FROM my-first-project-466601.modulabs_project.data
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 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 my-first-project-466601.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(Quantity) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS avg_quantity
FROM my-first-project-466601.modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
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 my-first-project-466601.modulabs_project.data AS
SELECT *
FROM my-first-project-466601.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 my-first-project-466601.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.84	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...
2	2011-01-18	541433	23166	-74215	2011-01-18 10:17:00 UTC	1.84	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...
3	2010-12-07	537626	22739	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BANGSLIVE ORA...
4	2010-12-07	537626	22739	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	LARGE HEART MEASURING SP...
5	2010-12-07	537626	71477	12	2010-12-07 14:57:00 UTC	3.25	12347	Iceland	COLOUR GLASS STAR TLAUGHT...
6	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB ACRYLIC...
7	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland	MINI PAINT SET VINTAGE
8	2010-12-07	537626	84967C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	BLUE 3 PRICE POLKA DOT CUTL...
9	2010-12-07	537626	22772	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	PINK DRAWER KNOB ACRYLIC...

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  (SELECT MAX(DATE(InvoiceDate)) FROM my-first-project-466601.modulabs_project.data) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM my-first-project-466601.modulabs_project.data;
```



```
SELECT MAX(InvoiceDate) AS most_recent_date
FROM my-first-project-466601.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

row	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-12-09	2011-01-18	541481	28166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE...
2	2011-12-09	2011-01-18	0541483	28166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE...
3	2011-12-09	2010-12-07	537626	22729	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BASELINE DBA...
4	2011-12-09	2010-12-07	537626	22729	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	LARGE HEART MEASURING SP...
5	2011-12-09	2010-12-07	537626	71477	12	2010-12-07 14:57:00 UTC	3.25	12347	Iceland	COLOUR GLASS STAR TIGHT...
6	2011-12-09	2010-12-07	537626	22855	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB ACRYLIC...
7	2011-12-09	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland	MINI PRINT SALT VINTAGE...
8	2011-12-09	2010-12-07	537626	809910	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	BLUE 3 PIECE POLYACRYLIC OUTL...

행	most_recent_date
1	2011-12-09

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(InvoiceDate) AS most_recent_InvoiceDay
FROM my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
ORDER BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	most_recent_Invo...
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
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17

- 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산하기

```
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDate) OVER () - InvoiceDate) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(InvoiceDate) AS InvoiceDate
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE my-first-project-466601.modulabs_project.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDate) OVER () - InvoiceDate) AS recency
FROM (
  SELECT
```

```

CustomerID,
MAX(InvoiceDate) AS InvoiceDay
FROM my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
)
ORDER BY recency;

```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

행	CustomerID	recency
1	16558	0
2	17490	0
3	15344	0
4	15804	0
5	14051	0
6	17001	0
7	17389	0
8	18102	0
9	15694	0
10	15910	0

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```

SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
ORDER BY CustomerID;

```

[결과 이미지를 넣어주세요]

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
8	12354	1
9	12355	1
10	12356	3

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```

SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
ORDER 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	1679

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

```
CREATE OR REPLACE TABLE my-first-project-466601.modulabs_project.user_rf AS
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY CustomerID
  ORDER BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY CustomerID
  ORDER 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 my-first-project-466601.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

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	14569	1	79	1
4	13436	1	76	1
5	13298	1	96	1
6	14204	1	72	2
7	15471	1	256	2
8	15195	1	1404	2
9	16569	1	93	3
10	16528	1	171	3

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM my-first-project-466601.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 my-first-project-466601.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 my-first-project-466601.modulabs_project.user_rf rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.6	794.6
2	15520	1	314	1	343.5	343.5
3	14569	1	79	1	227.4	227.4
4	13436	1	76	1	196.9	196.9
5	13298	1	96	1	360.0	360.0
6	14204	1	72	2	150.6	150.6
7	15471	1	256	2	454.5	454.5
8	15195	1	1404	2	3861.0	3861.0
9	16569	1	93	3	124.2	124.2
10	16528	1	171	3	244.4	244.4

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM my-first-project-466601.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.6	794.6
2	13298	1	96	1	360.0	360.0
3	13436	1	76	1	196.9	196.9
4	15520	1	314	1	343.5	343.5
5	14569	1	79	1	227.4	227.4
6	14204	1	72	2	150.6	150.6
7	15471	1	256	2	454.5	454.5
8	15195	1	1404	2	3861.0	3861.0
9	12650	1	250	3	242.4	242.4
10	15992	1	17	3	42.0	42.0

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 my-first-project-466601.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM my-first-project-466601.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	12713	1	505	0	794.6	794.6	37
2	15520	1	314	1	343.5	343.5	18
3	13298	1	96	1	360.0	360.0	2
4	13436	1	76	1	196.9	196.9	12
5	14569	1	79	1	227.4	227.4	10
6	14204	1	72	2	150.6	150.6	36
7	15471	1	256	2	454.5	454.5	67
8	15195	1	1404	2	3861.0	3861.0	1
9	14578	1	240	3	168.6	168.6	24
10	12478	1	233	3	546.0	546.0	35

2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 군 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]

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 my-first-project-466601.modulabs_project.user_data AS
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNT(CASE WHEN Quantity < 0 THEN 1 END) AS cancel_frequency
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY CustomerID
)
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate
FROM my-first-project-466601.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

일	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16953	1	10	30	30.8	20.8	1	0.0	1	0	0.0
2	13703	1	10	318	99.5	99.5	1	0.0	1	0	0.0
3	13829	1	-12	359	-102.0	-102.0	1	0.0	1	1	1.0
4	16257	1	1	176	21.9	21.9	1	0.0	1	0	0.0
5	16765	1	4	294	34.0	34.0	1	0.0	1	0	0.0
6	13017	1	48	7	204.0	204.0	1	0.0	1	0	0.0
7	16323	1	50	196	207.5	207.5	1	0.0	1	0	0.0
8	13188	1	24	11	99.6	99.6	1	0.0	1	0	0.0
9	17948	1	144	147	358.6	358.6	1	0.0	1	0	0.0
10	14119	1	-2	354	-19.9	-19.9	1	0.0	1	1	1.0

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

```
SELECT *
FROM my-first-project-466601.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

명	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16953	1	10	30	20.8	20.8	1	0.0	1	0	0.0
2	13703	1	10	318	99.5	99.5	1	0.0	1	0	0.0
3	13829	1	-12	359	-102.0	-102.0	1	0.0	1	1	1.0
4	16257	1	1	176	21.9	21.9	1	0.0	1	0	0.0
5	16765	1	4	294	34.0	34.0	1	0.0	1	0	0.0
6	13017	1	48	7	204.0	204.0	1	0.0	1	0	0.0
7	16323	1	50	196	207.5	207.5	1	0.0	1	0	0.0
8	13188	1	24	11	99.6	99.6	1	0.0	1	0	0.0
9	17948	1	144	147	358.6	358.6	1	0.0	1	0	0.0
10	14119	1	-2	354	-19.9	-19.9	1	0.0	1	1	1.0

회고

[회고 내용을 작성해주세요]

Keep : 다양한 USER FEATURE를 통해서 유저 분석이 가능하도록 데이터를 전처리 할 수 있다.

Problem : 실수로 조건을 잘못 입력해서 CREAT OR REPLACE TABLE 하면 고생한다..ㅠㅠ

Try : 다양한 소스들을 통해 SQL 쿼리 연습해야겠다.