

고객을 세그먼테이션하자! [프로젝트] - 나희정

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
SELECT *
FROM `verdant-bond-470202-k2.modulabs_project.data`
LIMIT 10
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	Invo
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2011
2	536365	71053	WHITE METAL LANTERN	6	2011
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2011
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2011
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2011
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2011
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2011
8	536366	22633	HAND WARMER UNION JACK	6	2011
9	536366	22632	HAND WARMER RED POLKA DOT	6	2011
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2011

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
SELECT COUNT(*) as row_count
FROM `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	row_count
1	541909

데이터 수 세기

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

```
# [[YOUR QUERY]]
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 `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	COUNT_UnitPrici
1	541909	541909	540455	541909	541909	5419

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

```
# [[YOUR QUERY]]

WITH data AS (
  SELECT * FROM `verdant-bond-470202-k2.modulabs_project.data`
)

SELECT 'InvoiceNo' AS column_name,
       COUNT(*) - COUNT(data.InvoiceNo) AS null_count,
       SAFE_DIVIDE(COUNT(*) - COUNT(data.InvoiceNo), COUNT(*)) AS null_ratio
FROM data

UNION ALL

SELECT 'StockCode',
       COUNT(*) - COUNT(data.StockCode),
       SAFE_DIVIDE(COUNT(*) - COUNT(data.StockCode), COUNT(*))
FROM data

UNION ALL

SELECT 'Description',
       COUNT(*) - COUNT(data.Description),
       SAFE_DIVIDE(COUNT(*) - COUNT(data.Description), COUNT(*))
FROM data

UNION ALL

SELECT 'Quantity',
       COUNT(*) - COUNT(data.Quantity),
       SAFE_DIVIDE(COUNT(*) - COUNT(data.Quantity), COUNT(*))
FROM data

UNION ALL

SELECT 'InvoiceDate',
       COUNT(*) - COUNT(data.InvoiceDate),
       SAFE_DIVIDE(COUNT(*) - COUNT(data.InvoiceDate), COUNT(*))
FROM data

UNION ALL

SELECT 'UnitPrice',
       COUNT(*) - COUNT(data.UnitPrice),
       SAFE_DIVIDE(COUNT(*) - COUNT(data.UnitPrice), COUNT(*))
FROM data

UNION ALL

SELECT 'CustomerID',
       COUNT(*) - COUNT(data.CustomerID),
       SAFE_DIVIDE(COUNT(*) - COUNT(data.CustomerID), COUNT(*))
FROM data

UNION ALL

SELECT 'Country',
       COUNT(*) - COUNT(data.Country),
```

```
SAFE_DIVIDE(COUNT(*) - COUNT(data.Country), COUNT(*))
FROM data;
```

[결과 이미지를 넣어주세요]

행	column_name	null_count	null_ratio
1	UnitPrice	0	0.0
2	Country	0	0.0
3	StockCode	0	0.0
4	InvoiceDate	0	0.0
5	CustomerID	135080	0.249266943342...
6	Quantity	0	0.0
7	Description	1454	0.002683107311...
8	InvoiceNo	0	0.0

결측치 처리 전략

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

```
SELECT # [YOUR QUERY]
FROM project_name.modulabs_project.data
# [YOUR QUERY];
SELECT StockCode, Description
FROM `verdant-bond-470202-k2.modulabs_project.data`
WHERE StockCode = '85123A'
GROUP BY StockCode, Description;
```

[결과 이미지를 넣어주세요]

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

결측치 처리

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

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

[결과 이미지를 넣어주세요]

작업 정보 **결과** 실행 세부정보 실행 그래프

i 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, **COUNT**가 1보다 큰 데이터를 세어보기

```
SELECT *
FROM project_name.modulabs_project.data
```

```
# [[YOUR QUERY]]
SELECT COUNT(*) AS duplicate_rows
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS cnt
  FROM `verdant-bond-470202-k2.modulabs_project.data`
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  HAVING COUNT(*) > 1
);
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화
행	duplicate_rows	
1	4837	

중복값 처리

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

```
# [[YOUR QUERY]];
CREATE OR REPLACE TABLE `verdant-bond-470202-k2.modulabs_project.data` AS
SELECT DISTINCT *
FROM `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
작업 탐색기			
<p>i 이 문으로 이름이 data인 테이블이 교체되었습니다.</p>			

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 **InvoiceNo**의 개수를 출력하기

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON
행	unique_invoice_c...		
1	22190		

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
# [[YOUR QUERY]]
SELECT DISTINCT InvoiceNo AS unique_invoice_count
FROM `verdant-bond-470202-k2.modulabs_project.data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JS
행	unique_invoice_count		
1	541431		
2	C541433		
3	537626		
4	542237		
5	549222		
6	556201		
7	562032		

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

```
SELECT *
FROM `verdant-bond-470202-k2.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description		
1	C541433	23166	MEDIUM CERAMIC TOP STORA...		
2	C545329	M	Manual		
3	C545329	M	Manual		
4	C545330	M	Manual		
5	C547388	22784	LANTERN CREAM GAZEBO		
6	C547388	37448	CERAMIC CAKE DESIGN SPOTT...		

- 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
) AS canceled_ratio_percent
FROM verdant-bond-470202-k2.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행
행	canceled_ratio_percent			
1			2.2	

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON
행	unique_stockcod...		
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 project_name.modulabs_project.data
# [[YOUR QUERY]] GROUP BY StockCode
ORDER BY sell_cnt DESC
# [[YOUR QUERY]]LIMIT 10;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보
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 project_name.modulabs_project.data
)
WHERE (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) BETWEEN 0 AND 1;

```

[결과 이미지를 넣어주세요]

메타데이터 선택				
작업 정보				
결과				
시각화				
JSON				
실행 세부정보				
실행 그래프				
행	StockCode	number_count	char_length	
1	POST	0	4	
2	M	0	1	
3	C2	1	1	
4	D	0	1	
5	BANK CHARGES	0	12	
6	PADS	0	4	
7	DOT	0	3	
8	CRUK	0	4	

- **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 DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM verdant-bond-470202-k2.modulabs_project.data
)
WHERE number_count BETWEEN 0 AND 1;

WITH ValidCodes AS (
  SELECT DISTINCT StockCode
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM `verdant-bond-470202-k2.modulabs_project.data`
)
WHERE number_count BETWEEN 0 AND 1
)
SELECT
  ROUND(
    COUNTIF(StockCode IN (SELECT StockCode FROM ValidCodes)) / COUNT(*) * 100,
    2
  ) AS ratio_percent
FROM `verdant-bond-470202-k2.modulabs_project.data`;

```


[결과 이미지를 넣어주세요]

작업 정보		
결과		
시각화		
재해 복구		
ratio_percent		
1	0.48	

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `verdant-bond-470202-k2.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 `verdant-bond-470202-k2.modulabs_project.data`
  )
  WHERE number_count BETWEEN 0 AND 1
);
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 data의 행 1,915개가 삭제되었습니다. </div>			

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM `verdant-bond-470202-k2.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```


[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보
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		

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM project_name.modulabs_project.data
WHERE UPPER(Description) IN ('NEXT DAY CARRIAGE', 'HIGH RESOLUTION IMAGE');
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 data의 행 83개가 삭제되었습니다. </div>			

- 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]

작업 정보 **결과** 실행 세부정보 실행 그래프

i 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 `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보 **결과** 시각화 JSON 실행 세부정보

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(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM `verdant-bond-470202-k2.modulabs_project.data`
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]


작업 정보 **결과** 시각화 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 project_name.modulabs_project.data AS
SELECT *
FROM project_name.modulabs_project.data
WHERE UnitPrice <> 0;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 이름이 data인 테이블이 교체되었습니다. </div>			

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	InvoiceDay	InvoiceNo	StockCode		
1	2011-01-18	541431	23166		
2	2011-01-18	C541433	23166		
3	2010-12-07	537626	84969		
4	2010-12-07	537626	84997D		

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  MAX(DATE(InvoiceDate)) OVER() AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `verdant-bond-470202-k2.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	most_recent_date	InvoiceDay	InvoiceNo		
1	2011-12-09	2011-09-28	568699		
2	2011-12-09	2011-11-06	574740		
3	2011-12-09	2011-10-13	571034		
4	2011-12-09	2011-08-19	563749		

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `verdant-bond-470202-k2.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON
행	CustomerID	InvoiceDay	
1	12346	2011-01-18	
2	12347	2011-12-07	
3	12348	2011-09-25	
4	12349	2011-11-21	

- 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 `verdant-bond-470202-k2.modulabs_project.data`
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON
행	CustomerID	recency	
1	12518	0	
2	12552	39	
3	12754	235	
4	13273	113	

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

```
CREATE OR REPLACE TABLE `verdant-bond-470202-k2.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 `verdant-bond-470202-k2.modulabs_project.data`
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

작업 정보
결과
실행 세부정보
실행 그래프

❗ 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `verdant-bond-470202-k2.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	시각화	JSON
행	CustomerID	purchase_cnt		
1	12346	2		
2	12347	7		
3	12348	4		
4	12349	1		

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `verdant-bond-470202-k2.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	시각화	JSON
행	CustomerID	item_cnt		
1	12346	0		
2	12347	2458		
3	12348	2332		
4	12349	630		

- 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 (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `verdant-bond-470202-k2.modulabs_project.data`
  GROUP BY CustomerID
),
```

```
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `verdant-bond-470202-k2.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 project_name.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
FROM `verdant-bond-470202-k2.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	시각화	JSON	실행
행	CustomerID ▼	user_total ▼			
1	12346	0.0			
2	12347	4310.0			
3	12348	1437.2			
4	12349	1457.6			

- 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 `verdant-bond-470202-k2.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 `verdant-bond-470202-k2.modulabs_project.user_rf` rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
  FROM `verdant-bond-470202-k2.modulabs_project.data`
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;

```

[결과 이미지를 넣어주세요]

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```

SELECT *
FROM `verdant-bond-470202-k2.modulabs_project.user_rfm`
LIMIT 100;

```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt	item_cnt	recency	user_tot
1	12713	1	505	0	
2	13436	1	76	1	
3	14569	1	79	1	
4	13298	1	96	1	

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]

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 이름이 user_data인 새 테이블이 생성되었습니다. </div>			


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]

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 이름이 user_data인 테이블이 교체되었습니다. </div>			

3. 구매 취소 경향성

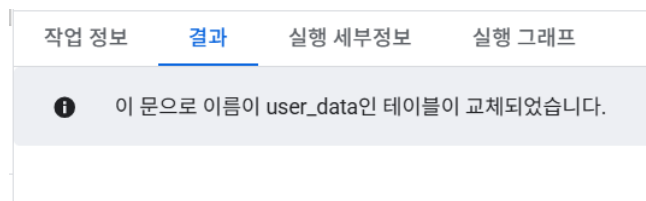
- 고객의 취소 패턴 파악하기
 - 취소 빈도(`cancel_frequency`) : 고객 별로 취소한 거래의 총 횟수
 - 취소 비율(`cancel_rate`) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
 - 취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기
(취소 비율은 소수점 두번째 자리)

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

```
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    # [[YOUR QUERY]] AS total_transactions,
    # [[YOUR QUERY]] AS cancel_frequency
  FROM project_name.modulabs_project.data
  # [[YOUR QUERY]]
)
```

```
SELECT u.*, t.* EXCEPT(CustomerID), # [[YOUR QUERY]] AS cancel_rate
FROM `project_name.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]



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

```
# [[YOUR QUERY]];
SELECT *
FROM `verdant-bond-470202-k2.modulabs_project.user_data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt	item_cnt	recency	user_total
1	13130	1	32	94	64.0
2	14962	1	44	270	126.7
3	18249	1	128	17	95.3
4	18209	1	58	28	139.1
5	14693	1	34	264	172.9

회고

[회고 내용을 작성해주세요]

Keep :

Problem : 정신이 혼미하다.

Try :