김주아_고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM learned-acolyte-482802-b5.modulabs_project.data
LIMIT 10
```

[결과 이미지]

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
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(*) AS total_rows
FROM learned-acolyte-482802-b5.modulabs_project.data
```

[결과 이미지]

total_rows
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 learned-acolyte-482802-b5.modulabs_project.data
```

[결과 이미지]

COUNT_InvoiceNo	COUNT_StockCode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	COUNT_UnitPrice	COUNT_Custome...	COUNT_Country
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 learned-acolyte-482802-b5.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) A
S missing_percentage
FROM learned-acolyte-482802-b5.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 learned-acolyte-482802-b5.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 mi
ssing_percentage
FROM learned-acolyte-482802-b5.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 learned-acolyte-482802-b5.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 m
issing_percentage
FROM learned-acolyte-482802-b5.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 learned-acolyte-482802-b5.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 mis
sing_percentage
FROM learned-acolyte-482802-b5.modulabs_project.data
```

[결과 이미지]

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM learned-acolyte-482802-b5.modulabs_project.data
WHERE StockCode = '85123A';
```

[결과 이미지]

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 learned-acolyte-482802-b5.modulabs_project.data
WHERE Description IS NULL OR CustomerID IS NULL;
```

[결과 이미지]

쿼리 결과 저장을 못해서 행 개수로 대체함

처리전/후(차이 135080)

total_rows
541909

total_rows
406829

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_cnt
FROM (
   SELECT InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
   FROM learned-acolyte-482802-b5.modulabs_project.data
   GROUP BY InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
   HAVING COUNT(*) > 1 )
```

[결과 이미지]

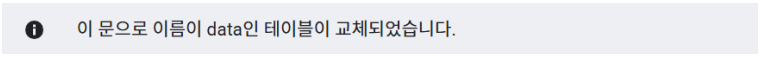
행	duplicate_cnt
1	4837

중복값 처리

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

```
CREATE OR REPLACE TABLE learned-acolyte-482802-b5.modulabs_project.data AS
SELECT DISTINCT *
FROM learned-acolyte-482802-b5.modulabs_project.data;
```

[결과 이미지]



행	total_rows
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice
FROM learned-acolyte-482802-b5.modulabs_project.data;
```

[결과 이미지]

행	unique_invoice
1	22190

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM learned-acolyte-482802-b5.modulabs_project.data
LIMIT 100;
```

[결과 이미지]

행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318
11	541998

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

```
SELECT *
FROM learned-acolyte-482802-b5.modulabs_project.data
WHERE InvoiceNo LIKE "C%"
LIMIT 100;
```

[결과 이미지]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
6	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
7	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
8	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
9	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway

- **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
FROM learned-acolyte-482802-b5.modulabs_project.data;
```

[결과 이미지]

행	canceled_ratio
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_StockCode
FROM learned-acolyte-482802-b5.modulabs_project.data;
```

[결과 이미지]

행	unique_StockCode
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 learned-acolyte-482802-b5.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지]

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 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 learned-acolyte-482802-b5.modulabs_project.data
)
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1;
```

[결과 이미지]

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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
SELECT ROUND(COUNTIF (
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1
      ) / COUNT(*) * 100, 2) AS out_ratio
FROM learned-acolyte-482802-b5.modulabs_project.data;
```

[결과 이미지]

행	out_ratio
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM learned-acolyte-482802-b5.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_cnt
    FROM learned-acolyte-482802-b5.modulabs_project.data
    )
  WHERE number_cnt <= 1 );
```

[결과 이미지]

> 이 문으로 data의 행 1,915개가 삭제되었습니다.

▼ Q. 이렇게 쓰면 안되나요?

```
DELETE FROM learned-acolyte-482802-b5.modulabs_project.data
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1;
```

Description 살펴보기

- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS description_cnt
FROM learned-acolyte-482802-b5.modulabs_project.data
GROUP BY Description
LIMIT 30;
```

[결과 이미지]

행	Description	description_cnt
1	MEDIUM CERAMIC TOP STORA...	208
2	MINI PAINT SET VINTAGE	335
3	ALARM CLOCK BAKELIKE PINK	636
4	RED TOADSTOOL LED NIGHT LI...	539
5	SET/3 DECOUPAGE STACKING ...	54
6	BATHROOM METAL SIGN	60
7	PURPLE DRAWERKNOB ACRYLI...	151

페이지당 결과 수: 50 1 – 30 (전체 30행)

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM learned-acolyte-482802-b5.modulabs_project.data
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```

[결과 이미지]

이 문으로 data의 행 83개가 삭제되었습니다.

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE learned-acolyte-482802-b5.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM learned-acolyte-482802-b5.modulabs_project.data;
```

[결과 이미지]

이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM learned-acolyte-482802-b5.modulabs_project.data;
```

[결과 이미지]

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 learned-acolyte-482802-b5.modulabs_project.data
WHERE UnitPrice = 0;
```

[결과 이미지]

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 learned-acolyte-482802-b5.modulabs_project.data AS
SELECT *
FROM learned-acolyte-482802-b5.modulabs_project.data
WHERE UnitPrice > 0 ;
```

[결과 이미지]

ⓘ 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- `InvoiceDate` **컬럼을 연월일 자료형으로 변경하기**

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM learned-acolyte-482802-b5.modulabs_project.data
```

[결과 이미지]

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...
3	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC	2.1	12347	Iceland	FOUR HOOK WHITE LOVEBIRDS
4	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLACK GRAND BAROQUE PHOT...
5	2010-12-07	537626	22775	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	PURPLE DRAWERKNOB ACRYLI...
6	2010-12-07	537626	21064	6	2010-12-07 14:57:00 UTC	5.95	12347	Iceland	BOOM BOX SPEAKER BOYS
7	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE GREEN
8	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	GREEN DRAWER KNOB ACRYLI...
9	2010-12-07	537626	22729	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE ORA...

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(DATE(InvoiceDate)) OVER() AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM learned-acolyte-482802-b5.modulabs_project.data
```

[결과 이미지]

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode
1	2011-12-09	2011-01-18	541431	23166
2	2011-12-09	2011-01-18	C541433	23166
3	2011-12-09	2010-12-07	537626	22212
4	2011-12-09	2010-12-07	537626	85167B
5	2011-12-09	2010-12-07	537626	22775
6	2011-12-09	2010-12-07	537626	21064
7	2011-12-09	2010-12-07	537626	22726
8	2011-12-09	2010-12-07	537626	22773

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

```
SELECT CustomerID, MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM learned-acolyte-482802-b5.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지]

행	CustomerID	InvoiceDay
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

- **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 learned-acolyte-482802-b5.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지]

행	CustomerID	recency
1	12592	92
2	12635	89
3	12732	179
4	13047	46
5	13140	108
6	13144	332
7	13258	11
8	13318	1
9	13479	198

▼ Q. 이렇게 쓰면 안되나요?

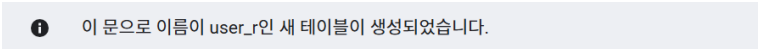
```
DATE_DIFF(MAX(InvoiceDay) OVER(), InvoiceDay, DAY) AS recency
```

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

```
CREATE OR REPLACE TABLE learned-acolyte-482802-b5.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 learned-acolyte-482802-b5.modulabs_project.data
```

```
  GROUP BY CustomerID
);
```

[결과 이미지]



행	CustomerID	recency
1	12423	0
2	12713	0
3	15910	0
4	15694	0
5	12518	0
6	12526	0
7	17754	0
8	16626	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM learned-acolyte-482802-b5.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지]

행	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84
7	12353	4
8	12354	58

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM learned-acolyte-482802-b5.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지]

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

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

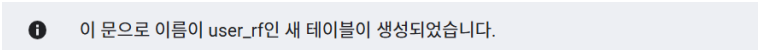
```
CREATE OR REPLACE TABLE learned-acolyte-482802-b5.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS ( SELECT CustomerID, COUNT(InvoiceNo) AS purchase_cnt
   FROM learned-acolyte-482802-b5.modulabs_project.data
   GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS ( SELECT CustomerID, SUM(Quantity) AS item_cnt
   FROM learned-acolyte-482802-b5.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 learned-acolyte-482802-b5.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지]



작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency
1	13777	217	12807	0
2	17001	169	2164	0
3	17389	223	7442	0
4	12518	119	1306	0
5	14446	276	856	0
6	12423	118	1312	0
7	14422	222	2906	0
8	12526	68	624	0
9	17754	90	1767	0

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
```

```
FROM learned-acolyte-482802-b5.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지]

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

- **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 learned-acolyte-482802-b5.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 learned-acolyte-482802-b5.modulabs_project.user_rf rf
LEFT JOIN (
    -- 고객 별 총 지출액
  SELECT
      CustomerID, ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
    FROM learned-acolyte-482802-b5.modulabs_project.data
        GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지]

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM learned-acolyte-482802-b5.modulabs_project.user_rfm;
```

[결과 이미지]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12423	118	1312	0	1624.1	13.8
2	17490	85	1022	0	1936.8	22.8
3	12748	4440	23516	0	29820.0	6.7
4	12526	68	624	0	1172.7	17.2
5	16626	184	2670	0	4379.7	23.8
6	15804	273	2513	0	3848.5	14.1
7	12518	119	1306	0	1840.9	15.5
8	13069	469	5454	0	3713.1	7.9
9	17389	223	7442	0	31317.5	140.4
10	14422	222	2906	0	4263.6	19.2

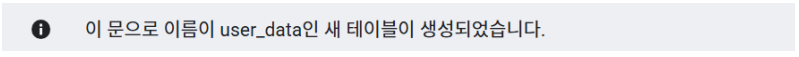
11-8. 추가 Feature 추출

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 이미지]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	18133	1	1350	212	931.5	931.5	1
2	12791	1	96	373	177.6	177.6	1
3	16323	1	50	196	207.5	207.5	1
4	14705	1	100	198	179.0	179.0	1
5	16990	1	100	218	179.0	179.0	1
6	16738	1	3	297	3.8	3.8	1
7	13270	1	200	366	590.0	590.0	1
8	16995	1	-1	372	-1.3	-1.3	1
9	13135	1	4300	196	3096.0	3096.0	1

2. 평균 구매 주기

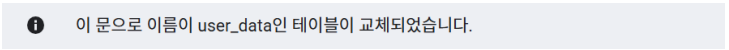
- **고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)**
  - **평균 구매 소요 일수를 계산하고, 그 결과를** `user_data` **에 통합**

```
CREATE OR REPLACE TABLE learned-acolyte-482802-b5.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
      learned-acolyte-482802-b5.modulabs_project.data
```

```
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM learned-acolyte-482802-b5.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	15313	1	25	110	52.0	52.0	1	0.0
2	15668	1	72	217	76.3	76.3	1	0.0
3	16737	1	288	53	417.6	417.6	1	0.0
4	15510	1	2	330	250.0	250.0	1	0.0
5	14119	1	-2	354	-19.9	-19.9	1	0.0
6	18113	1	72	368	76.3	76.3	1	0.0
7	17925	1	72	372	244.1	244.1	1	0.0
8	14705	1	100	198	179.0	179.0	1	0.0
9	16765	1	4	294	34.0	34.0	1	0.0

3. 구매 취소 경향성

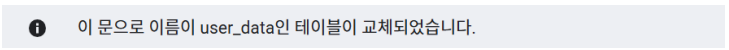
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE learned-acolyte-482802-b5.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    -- 전체 거래 횟수
    COUNT(InvoiceNo) AS total_transactions,
    -- 1) 취소 빈도 (InvoiceNo가 'C'로 시작하는 것만 골라서 세기)
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS cancel_frequency
  FROM learned-acolyte-482802-b5.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID),
    -- 2) 취소 비율 계산 (취소 횟수 / 전체 횟수)
    ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate
FROM learned-acolyte-482802-b5.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지]



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

```
SELECT *
FROM learned-acolyte-482802-b5.modulabs_project.user_data;
```

[결과 이미지]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	14351	1	12	164	51.0	51.0	1	0.0	1	0	0.0
2	17752	1	192	359	80.6	80.6	1	0.0	1	0	0.0
3	17986	1	10	56	20.8	20.8	1	0.0	1	0	0.0
4	16737	1	288	53	417.6	417.6	1	0.0	1	0	0.0
5	17925	1	72	372	244.1	244.1	1	0.0	1	0	0.0
6	15070	1	36	372	106.2	106.2	1	0.0	1	0	0.0
7	16953	1	10	30	20.8	20.8	1	0.0	1	0	0.0
8	13391	1	4	203	59.8	59.8	1	0.0	1	0	0.0
9	15940	1	4	311	35.8	35.8	1	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep :

- 이해가 안되는 부분을 그냥 넘어가지 않고 이유를 알아내려고 노력함
- 먼저 하고서 수정하는 방식이 아닌 제대로된 쿼리문을 작성할 수 있도록 여러번 확인함
- RFM 분석과 추가적인 특징들을 통해서 데이터를 뽑아내는 과정
- 포기하지 않고 끝까지 쿼리 완성

Problem :

- 에러가 났을 때 원인을 빠르게 파악하지 못하고 당황함
- 문법은 이해가 되는데 서브쿼리가 들어가면 아직 로직이 빠르게 이해되지 않음
- 이게 왜 필요한지에 대한 큰 그림보다는 코드 짜는데 아직은 급급함

Try :

- 오늘 썼던 함수나 구문들을 복습해서 익힐 수 있도록 계속 써보기
- 에러가 났을 때 에러 메시지 천천히 읽으며 원인 파악하기
- 테이블을 통해 알 수 있는 내용 정리해보기
- 오늘 작성한 코드 다시 한번 쳐보면서 복기하기