고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM modular-bucksaw-439401-m3.modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]

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

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custo
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	0.0	
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	0.0	
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	0.0	

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM `modular-bucksaw-439401-m3.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 `modular-bucksaw-439401-m3.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 ALL을 통해 합치기**

```
SELECT 'InvoiceNo' AS column_name, COUNT(*) - COUNT(InvoiceNo) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
UNION ALL
SELECT 'StockCode' AS column_name, COUNT(*) - COUNT(StockCode) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
UNION ALL
SELECT 'Description' AS column_name, COUNT(*) - COUNT(Description) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
UNION ALL
SELECT 'Quantity' AS column_name, COUNT(*) - COUNT(Quantity) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
UNION ALL
SELECT 'InvoiceDate' AS column_name, COUNT(*) - COUNT(InvoiceDate) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
UNION ALL
SELECT 'UnitPrice' AS column_name, COUNT(*) - COUNT(UnitPrice) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
UNION ALL
SELECT 'CustomerID' AS column_name, COUNT(*) - COUNT(CustomerID) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
UNION ALL
SELECT 'Country' AS column_name, COUNT(*) - COUNT(Country) AS missing_values
FROM `modular-bucksaw-439401-m3.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	column_name	missing_values
1	InvoiceNo	0
2	Country	0
3	UnitPrice	0
4	InvoiceDate	0
5	Description	1454
6	StockCode	0
7	Quantity	0
8	CustomerID	135080

결측치 처리 전략

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

```
SELECT Description
FROM `modular-bucksaw-439401-m3.modulabs_project.data`
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

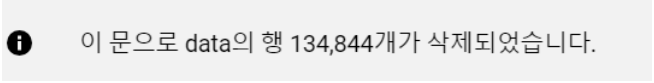
행	Description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	CREAM HANGING HEART T-LIG...
5	CREAM HANGING HEART T-LIG...
6	CREAM HANGING HEART T-LIG...
7	CREAM HANGING HEART T-LIG...
8	CREAM HANGING HEART T-LIG...

결측치 처리

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

```
# 85123A 부분만 먼저 제거해서 결과에 차이가 조금 있습니다.
DELETE FROM `modular-bucksaw-439401-m3.modulabs_project.data`
WHERE CustomerID IS NULL OR Description IS NULL;
```

[결과 이미지를 넣어주세요]



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 InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
    FROM `modular-bucksaw-439401-m3.modulabs_project.data`
    GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Countr
    HAVING COUNT(*) > 1
) AS duplicate_rows;
```

[결과 이미지를 넣어주세요]

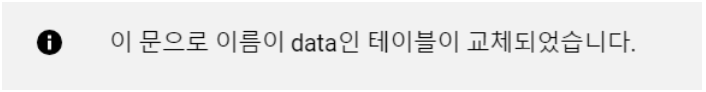
행	f0_
1	4837

중복값 처리

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

```
CREATE OR REPLACE TABLE modular-bucksaw-439401-m3.modulabs_project.data AS
SELECT DISTINCT *
FROM modular-bucksaw-439401-m3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

- **고유(unique)한 `InvoiceNo`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM modular-bucksaw-439401-m3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	unique_invoice_coun
1	22190

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

```
SELECT DISTINCT InvoiceNo
FROM modular-bucksaw-439401-m3.modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032
7	561387
8	574868

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

```
SELECT *
FROM modular-bucksaw-439401-m3.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Cu
1	C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	
2	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	
3	C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	
4	C554983	47590B	PINK HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	
5	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	
6	C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	
7	C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	
8	C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	

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

```
SELECT
    ROUND((COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 END) * 100.0) / COUNT(*), 1) AS canceled_perce
FROM modular-bucksaw-439401-m3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	canceled_percentage
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM modular-bucksaw-439401-m3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	unique_stockcode_c
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 modular-bucksaw-439401-m3.modulabs_project.data1
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

- `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 modular-bucksaw-439401-m3.modulabs_project.data1
)
WHERE number_count <= 1;
```

[결과 이미지를 넣어주세요]

행	StockCode	number_count
1	POST	0
2	M	0
3	PADS	0
4	D	0
5	BANK CHARGES	0
6	DOT	0
7	CRUK	0
8	C2	1

- `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((COUNT(*) * 100.0) / (SELECT COUNT(*) FROM modular-bucksaw-439401-m3.modulabs_project.data1)
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM modular-bucksaw-439401-m3.modulabs_project.data1
)
WHERE number_count <= 1;
```

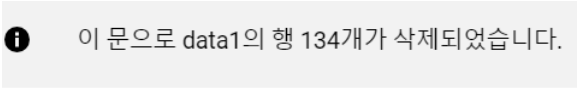
[결과 이미지를 넣어주세요]

행	percentage
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM modular-bucksaw-439401-m3.modulabs_project.data1
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM (
        SELECT StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
        FROM modular-bucksaw-439401-m3.modulabs_project.data1
         )
  WHERE number_count <= 1
    )
)
```

[결과 이미지를 넣어주세요]



`Description` 살펴보기

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

```
SELECT Description, COUNT(*) AS frequency
FROM modular-bucksaw-439401-m3.modulabs_project.data1
GROUP BY Description
ORDER BY frequency DESC
LIMIT 30;
```

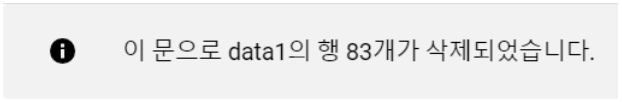
[결과 이미지를 넣어주세요]

행	Description	frequency
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
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE	1062

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE FROM modular-bucksaw-439401-m3.modulabs_project.data1
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```

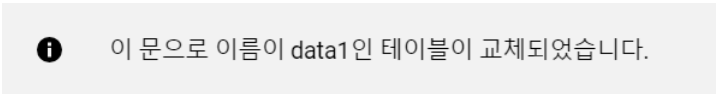
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE modular-bucksaw-439401-m3.modulabs_project.data1 AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM modular-bucksaw-439401-m3.modulabs_project.data1;
```

[결과 이미지를 넣어주세요]



`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 modular-bucksaw-439401-m3.modulabs_project.data1;
```

[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(*) AS zero_price_count,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM modular-bucksaw-439401-m3.modulabs_project.data1
WHERE UnitPrice = 0;
```

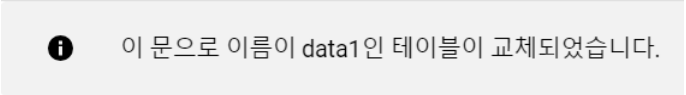
[결과 이미지를 넣어주세요]

행	zero_price_count	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

- `UnitPrice = 0`를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE modular-bucksaw-439401-m3.modulabs_project.data1 AS
SELECT *
FROM modular-bucksaw-439401-m3.modulabs_project.data1
WHERE UnitPrice != 0;
```

[결과 이미지를 넣어주세요]



11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM modular-bucksaw-439401-m3.modulabs_project.data1;
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID
1	2011-11-03	574301	22960	6	2011-11-03 16:15:00 UTC	4.25	1254
2	2011-11-03	574301	23511	6	2011-11-03 16:15:00 UTC	2.08	1254
3	2011-11-03	574301	23240	6	2011-11-03 16:15:00 UTC	4.15	1254
4	2011-11-03	574301	85049E	12	2011-11-03 16:15:00 UTC	1.25	1254
5	2011-11-03	574301	22621	12	2011-11-03 16:15:00 UTC	1.65	1254
6	2011-11-03	574301	22751	4	2011-11-03 16:15:00 UTC	3.75	1254
7	2011-11-03	574301	20749	4	2011-11-03 16:15:00 UTC	7.95	1254
8	2011-11-03	574301	22086	6	2011-11-03 16:15:00 UTC	2.95	1254

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
   MAX(InvoiceDate) OVER () AS most_recent_date,
   DATE(InvoiceDate) AS InvoiceDay,
   *
FROM modular-bucksaw-439401-m3.modulabs_project.data1;
```

[결과 이미지를 넣어주세요]

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-12-09 12:50:00 UTC	2011-06-19	557305	47574A	2	2011-06-19 14:42:00 UTC
2	2011-12-09 12:50:00 UTC	2011-10-14	571241	22307	2	2011-10-14 14:58:00 UTC
3	2011-12-09 12:50:00 UTC	2011-11-25	578781	84201B	1	2011-11-25 11:54:00 UTC
4	2011-12-09 12:50:00 UTC	2011-08-26	564647	23009	4	2011-08-26 13:56:00 UTC
5	2011-12-09 12:50:00 UTC	2011-12-02	580287	21034	1	2011-12-02 13:28:00 UTC
6	2011-12-09 12:50:00 UTC	2011-12-05	580647	23079	2	2011-12-05 13:14:00 UTC
7	2011-12-09 12:50:00 UTC	2011-10-19	571829	85152	48	2011-10-19 11:53:00 UTC
8	2011-12-09 12:50:00 UTC	2011-10-11	570593	22338	96	2011-10-11 11:38:00 UTC

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM modular-bucksaw-439401-m3.modulabs_project.data1
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	InvoiceDay
1	12544	2011-11-10
2	13568	2011-06-19
3	13824	2011-11-07
4	14080	2011-11-07
5	14336	2011-11-23
6	14592	2011-11-04
7	15104	2011-06-26
8	15360	2011-10-31

- **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 project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

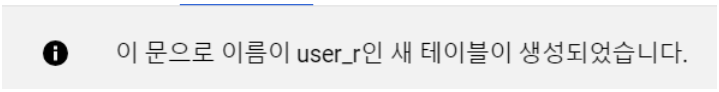
행	CustomerID	recency
1	15632	15
2	16657	9
3	15394	9
4	13870	2
5	13871	22
6	17456	365
7	14133	129
8	16696	135

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

```
CREATE OR REPLACE TABLE modular-bucksaw-439401-m3.modulabs_project.user_r AS
WITH recent_purchases AS (
```

```
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM modular-bucksaw-439401-m3.modulabs_project.data1
  GROUP BY CustomerID
),
most_recent_date AS (
  SELECT MAX(InvoiceDay) AS most_recent_date
  FROM recent_purchases
)
SELECT
  rp.CustomerID,
  rp.InvoiceDay,
  EXTRACT(DAY FROM (mr.most_recent_date - rp.InvoiceDay)) AS recency
FROM recent_purchases rp, most_recent_date mr;
```

[결과 이미지를 넣어주세요]



Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM modular-bucksaw-439401-m3.modulabs_project.data1
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
1	12544	2
2	13568	1
3	13824	5
4	14080	1
5	14336	4
6	14592	3
7	15104	3
8	15360	1

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM modular-bucksaw-439401-m3.modulabs_project.data1
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	item_cnt
1	12544	130
2	13568	66
3	13824	768
4	14080	48
5	14336	1759
6	14592	407
7	15104	633
8	15360	223
9	15872	197

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

```
CREATE OR REPLACE TABLE modular-bucksaw-439401-m3.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM modular-bucksaw-439401-m3.modulabs_project.data1
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM modular-bucksaw-439401-m3.modulabs_project.data1
  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 modular-bucksaw-439401-m3.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	15083	1	38	256
3	18010	1	60	256
4	12792	1	215	256
5	13436	1	76	1
6	13298	1	96	1
7	15520	1	314	1
8	14569	1	79	1
9	14476	1	110	257
10	13357	1	321	257
11	14204	1	72	2
12	15195	1	1404	2
13	15471	1	256	2
14	12442	1	181	3
15	15992	1	17	3
16	15318	1	642	3
17	16569	1	93	3
18	12478	1	233	3
19	14578	1	240	3
20	17914	1	457	3
21	12650	1	250	3
22	16528	1	171	3
23	14536	1	39	259

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM modular-bucksaw-439401-m3.modulabs_project.data1
GROUP BY CustomerID;
```

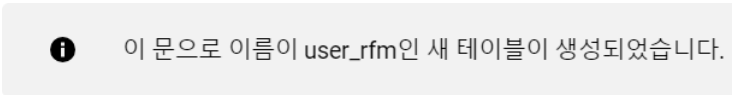
[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	12544	299.7
2	13568	187.0
3	13824	1698.9
4	14080	45.6
5	14336	1614.9
6	14592	557.9
7	15104	968.6
8	15360	427.9
9	15872	216.2

- **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 modular-bucksaw-439401-m3.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 modular-bucksaw-439401-m3.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM modular-bucksaw-439401-m3.modulabs_project.data1
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM modular-bucksaw-439401-m3.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	794.5
2	18010	1	60	256	174.8	174.8
3	15083	1	38	256	88.2	88.2
4	12792	1	215	256	344.5	344.5
5	15520	1	314	1	343.5	343.5
6	13298	1	96	1	360.0	360.0
7	14569	1	79	1	227.4	227.4
8	13436	1	76	1	196.9	196.9

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]

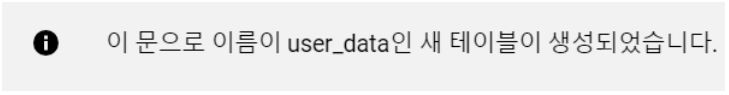
2. 평균 구매 주기

- **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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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

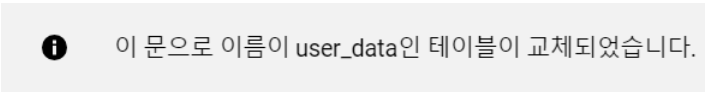
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
    - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
      (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE modular-bucksaw-439401-m3.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 END) AS cancel_frequency
  FROM modular-bucksaw-439401-m3.modulabs_project.data1
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID),
  ROUND(cancel_frequency * 100.0 / total_transactions, 2) AS cancel_rate
FROM `modular-bucksaw-439401-m3.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM `modular-bucksaw-439401-m3.modulabs_project.user_data`
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transact
1	16406	1	116	18	154.4	154.4	54	0.0	
2	15000	1	218	47	490.5	490.5	76	0.0	
3	14675	1	336	16	596.4	596.4	93	0.0	
4	12688	1	3028	113	4873.8	4873.8	171	0.0	
5	17619	1	124	312	214.4	214.4	71	0.0	
6	17614	1	235	57	386.3	386.3	97	0.0	
7	12371	1	582	59	1528.0	1528.0	62	0.23	
8	15019	1	150	266	252.8	252.8	64	0.0	