김지연_고객을 세그먼테이션하자 [프로젝트]

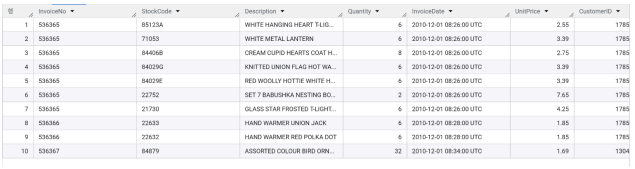
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM modulabs_project.data
LIMIT 10;
```

[결과 이미지]



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	1785
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	1785
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	1785
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	1785
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	1785
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	1785
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	1785
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	1785
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	1785
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	1304

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

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

[결과 이미지]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
41	536370	22900	SET 2 TEA TOWELS I LOVE LON...	24	2010-12-01 08:45:00 UTC	2.95	1[illegible]
42	536370	21913	VINTAGE SEASIDE JIGSAW PU...	12	2010-12-01 08:45:00 UTC	3.75	1[illegible]
43	536370	22540	MINI JIGSAW CIRCUS PARADE	24	2010-12-01 08:45:00 UTC	0.42	1[illegible]
44	536370	22544	MINI JIGSAW SPACEBOY	24	2010-12-01 08:45:00 UTC	0.42	1[illegible]
45	536370	22492	MINI PAINT SET VINTAGE	36	2010-12-01 08:45:00 UTC	0.65	1[illegible]
46	536370	POST	POSTAGE	3	2010-12-01 08:45:00 UTC	18.0	1[illegible]
47	536371	22086	PAPER CHAIN KIT 50'S CHRIST...	80	2010-12-01 09:00:00 UTC	2.55	1[illegible]
48	536372	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 09:01:00 UTC	1.85	1[illegible]
49	536372	22633	HAND WARMER UNION JACK	6	2010-12-01 09:01:00 UTC	1.85	1[illegible]
50	536373	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 09:02:00 UTC	2.55	1[illegible]

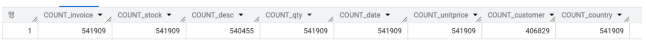
페이지당 결과 수: 50 1 – 50 (전체 541909행)

데이터 수 세기

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

```
SELECT
    COUNT(InvoiceNo) AS COUNT_invoice,
    COUNT(StockCode) AS COUNT_stock,
    COUNT(Description) AS COUNT_desc,
    COUNT(Quantity) AS COUNT_qty,
    COUNT(InvoiceDate) AS COUNT_date,
    COUNT(UnitPrice) AS COUNT_unitprice,
    COUNT(CustomerID) AS COUNT_customer,
    COUNT(Country) AS COUNT_country
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



행	COUNT_invoice	COUNT_stock	COUNT_desc	COUNT_qty	COUNT_date	COUNT_unitprice	COUNT_customer	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 column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(
   SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM modulabs_project.data UNION ALL
   SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM modulabs_project.data UNION ALL
   SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM modulabs_project.data UNION ALL
   SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM modulabs_project.data UNION ALL
   SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM modulabs_project.data UNION ALL
   SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM modulabs_project.data UNION ALL
   SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM modulabs_project.data UNION ALL
   SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	column_name	f0_
1	InvoiceNo	0.0
2	StockCode	0.0
3	Description	0.27
4	Quantity	0.0
5	InvoiceDate	0.0
6	UnitPrice	0.0
7	CustomerID	24.93
8	Country	0.0

결측치 처리 전략

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

```
SELECT DISTINCT(Description)
FROM modulabs_project.data
WHERE StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

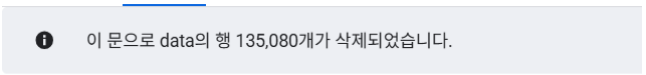
행	Description
1	WHITE HANGING HEART T-LIG...
2	?
3	wrongly marked carton 22804
4	CREAM HANGING HEART T-LIG...

결측치 처리

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

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

[결과 이미지를 넣어주세요]



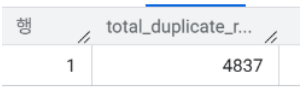
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- **중복된 행의 수를 세어보기**
  - **8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기**

```
SELECT COUNT(*) AS total_duplicate_rows
FROM (

SELECT COUNT(*) AS duplicate_cnt

FROM modulabs_project.data

GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country

HAVING COUNT(*) > 1

) AS sub;
```

[결과 이미지를 넣어주세요]



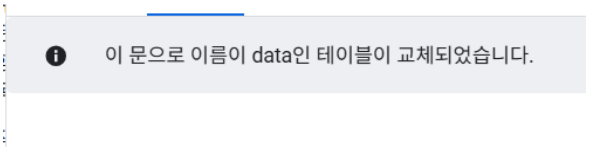
행	total_duplicate_r...
1	4837

중복값 처리

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

```
CREATE OR REPLACE TABLE `project-40efe70e-5db7-4fab-948.modulabs_project.data` AS
SELECT DISTINCT *
FROM `project-40efe70e-5db7-4fab-948.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	22190

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

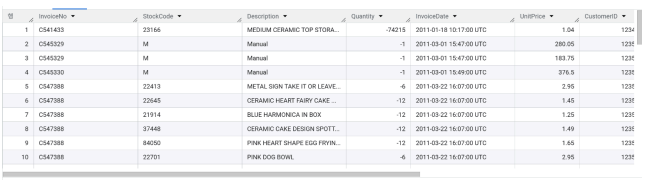
행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT *
FROM modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]



- **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)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

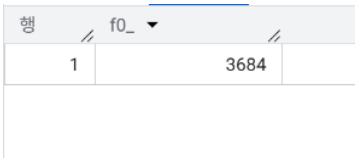
행	f0_
1	2.2

StockCode 살펴보기

- **고유한 StockCode의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode)
FROM 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 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

- **StockCode의 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 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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
SELECT DISTINCT ROUND(SUM(CASE WHEN number_count IN (0,1) THEN 1 ELSE 0 END)/COUNT(*) *100, 2)
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM modulabs_project.data
)
```

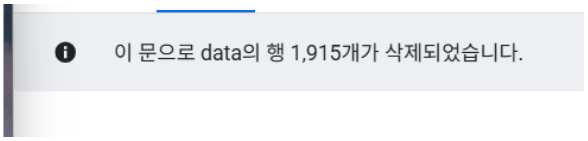
[결과 이미지를 넣어주세요]

행	f0_
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM
    (
      SELECT StockCode
      FROM modulabs_project.data
      WHERE  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0,1)

    )
);
```

[결과 이미지를 넣어주세요]



Description 살펴보기

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

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

[결과 이미지를 넣어주세요]

행	Description	description_cnt
21	WOODEN PICTURE FRAME WHI...	900
22	JUMBO BAG PINK POLKADOT	897
23	SET OF 4 PANTRY JELLY MOUL...	890
24	LUNCH BAG APPLE DESIGN	890
25	BAKING SET 9 PIECE RETROSP...	885
26	RECIPE BOX PANTRY YELLOW ...	883
27	JAM MAKING SET PRINTED	883
28	LUNCH BAG WOODLAND	850
29	ROSES REGENCY TEACUP AND ...	844
30	VICTORIAN GLASS HANGING T...	843

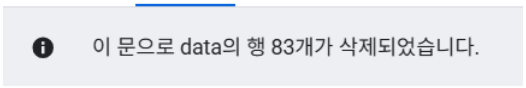
페이지당 결과 수: 50 1 – 30 (전체 30행)

작업 기록

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM project_name.modulabs_project.data
WHERE
Description IN ('Next Day Carriage', 'High Resolution Image')
```

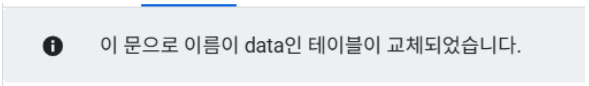
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM 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(*) AS cnt_quantity, MIN(Quantity) AS min_quantity,MAX(Quantity) AS max_quantity, AVG(Quantity) AS avg_q
uantity
FROM modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

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

- **UnitPrice = 0를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT *
FROM modulabs_project.data
WHERE UnitPrice > 0
```

[결과 이미지를 넣어주세요]

ⓘ 이 문으로 이름이 data인 테이블이 교체되었습니다.

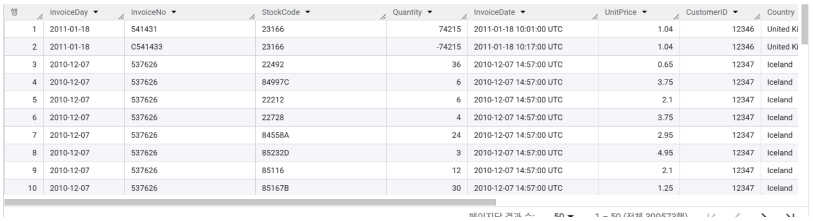
11-7. RFM 스코어

Recency

- **InvoiceDate 컬럼을 연월일 자료형으로 변경하기**

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

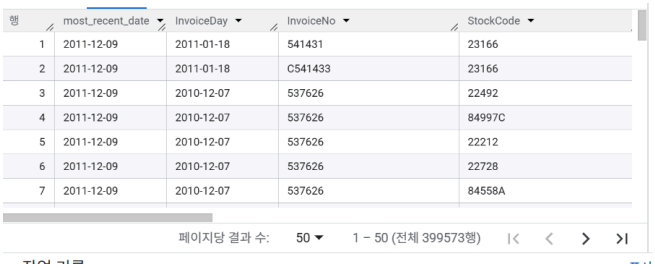


행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Ki
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Ki
3	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland
4	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
5	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC	2.1	12347	Iceland
6	2010-12-07	537626	22728	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
7	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC	2.95	12347	Iceland
8	2010-12-07	537626	85232D	3	2010-12-07 14:57:00 UTC	4.95	12347	Iceland
9	2010-12-07	537626	85116	12	2010-12-07 14:57:00 UTC	2.1	12347	Iceland
10	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(DATE(InvoiceDate)) OVER () AS most_recent_date, -- 데이터 전체에서 가장 큰(최근) 날짜
  DATE(InvoiceDate) AS InvoiceDay,          -- 시분초를 제외한 구매 날짜만 추출
  *
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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	22492
4	2011-12-09	2010-12-07	537626	84997C
5	2011-12-09	2010-12-07	537626	22212
6	2011-12-09	2010-12-07	537626	22728
7	2011-12-09	2010-12-07	537626	84558A

페이지당 결과 수: 50 1 – 50 (전체 399573행)

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

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

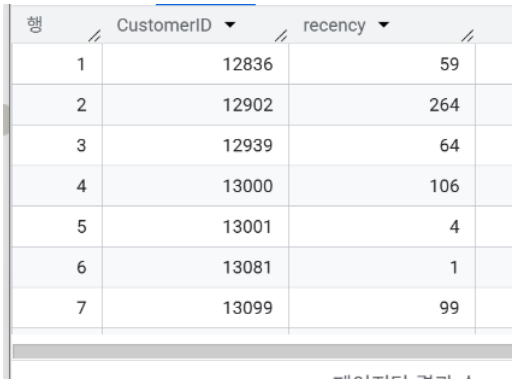
행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19

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

[결과 이미지를 넣어주세요]

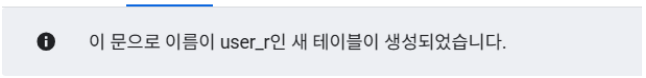


행	CustomerID	recency
1	12836	59
2	12902	264
3	12939	64
4	13000	106
5	13001	4
6	13081	1
7	13099	99

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

```
CREATE OR REPLACE TABLE `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 modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

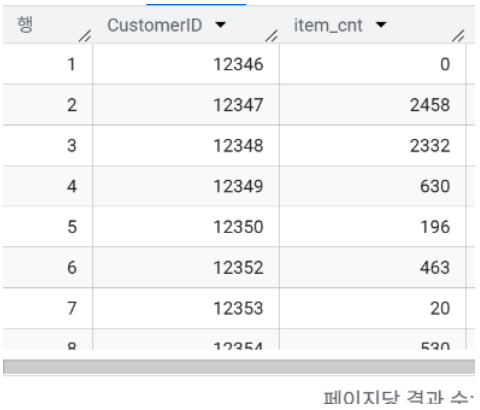
```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM modulabs_project.data
GROUP 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

페이지당 결과 수:

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

```
CREATE OR REPLACE TABLE modulabs_project.user_rf AS

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

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

[결과 이미지를 넣어주세요]

ⓘ 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice),0) AS user_total
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

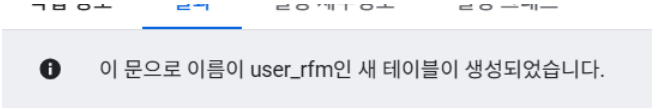
행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.0
4	12349	1458.0
5	12350	294.0
6	12352	1265.0
7	12353	89.0
8	12354	1079.0
9	12355	459.0

- **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**

```
CREATE OR REPLACE TABLE 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 modulabs_project.user_rf rf
LEFT JOIN (
```

```
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
  ROUND(SUM(Quantity * UnitPrice),0) AS user_total
FROM modulabs_project.data
GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

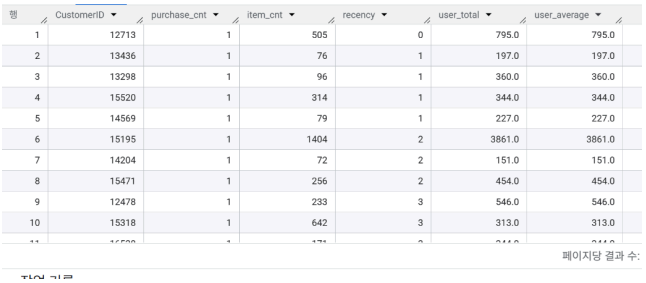


RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13436	1	76	1	197.0	197.0
3	13298	1	96	1	360.0	360.0
4	15520	1	314	1	344.0	344.0
5	14569	1	79	1	227.0	227.0
6	15195	1	1404	2	3861.0	3861.0
7	14204	1	72	2	151.0	151.0
8	15471	1	256	2	454.0	454.0
9	12478	1	233	3	546.0	546.0
10	15318	1	642	3	313.0	313.0

페이지당 결과 수:

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

[결과 이미지를 넣어주세요]

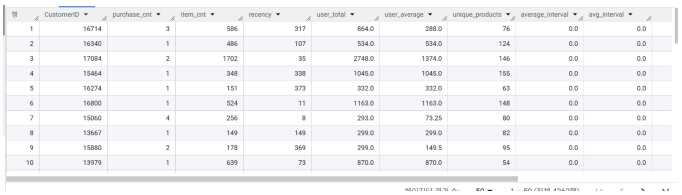
이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

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 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
      modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	avg_interval
1	16714	3	586	317	864.0	288.0	76	0.0	0.0
2	16340	1	486	107	534.0	534.0	124	0.0	0.0
3	17084	2	1702	35	2748.0	1374.0	146	0.0	0.0
4	15464	1	348	338	1045.0	1045.0	155	0.0	0.0
5	16274	1	151	373	332.0	332.0	63	0.0	0.0
6	16800	1	524	11	1163.0	1163.0	148	0.0	0.0
7	15060	4	256	8	293.0	73.25	80	0.0	0.0
8	13667	1	149	149	299.0	299.0	82	0.0	0.0
9	15880	2	178	369	299.0	149.5	95	0.0	0.0
10	13979	1	639	73	870.0	870.0	54	0.0	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 modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNTIF(InvoiceNo LIKE 'C%') AS cancel_frequency
```

```
  FROM modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND((cancel_frequency/total_transactions) *100,2) AS cancel_rate
FROM 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 modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

회고

[회고 내용을 작성해주세요]

- 고객을 새그맨테이션 하는 과정을 처음부터 끝까지 경험해볼 수 있어 매우 좋았습니다. 다만 아직 sql 쿼리문법에 익숙하지 않아 간단한 쿼리에도 몇몇 내용을 빼먹거나 오류가 나서 다시 하는 과정들이 많이 있었습니다. 온전히 제 힘으로 처음부터 끝까지 했다고 말하긴 어렵지만 그래도 전체 과정을 경험해보고, 실제 데이터 분석가들이 이런식으로 현업에서 일할 것이라는 대강의 그림을 그려볼 수 있어 좋았습니다. 서브쿼리문, window 구문에 대해 더 공부를 해야 할 것 같다고 생각합니다 ^^