

# 고객을 세그멘테이션하자 [프로젝트]

\*크기 60으로 내보내기

\*깃헙 레포지토리 업로드 후 제출

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
select *  
from oceanic-citadel-439400-c9.molab.data  
limit 10
```

입력 정보	결과	직접	JSON	실행 세부정보	실행 기록			
행	InvoiceNo	StockCode	Description	Quantity	Invoicedate	UnitPrice	CustomerID	Country
1	536414	22139	nafl	56	2010-12-01 11:52:00 UTC	0.0	nafl	United Kingdom
2	536545	21134	nafl	1	2010-12-01 14:32:00 UTC	0.0	nafl	United Kingdom
3	536546	22145	nafl	1	2010-12-01 14:33:00 UTC	0.0	nafl	United Kingdom
4	536547	37509	nafl	1	2010-12-01 14:33:00 UTC	0.0	nafl	United Kingdom
5	536549	85234A	nafl	1	2010-12-01 14:34:00 UTC	0.0	nafl	United Kingdom
6	536550	85044	nafl	1	2010-12-01 14:34:00 UTC	0.0	nafl	United Kingdom
7	536552	20950	nafl	1	2010-12-01 14:34:00 UTC	0.0	nafl	United Kingdom
8	536553	37461	nafl	3	2010-12-01 14:35:00 UTC	0.0	nafl	United Kingdom
9	536554	84670	nafl	23	2010-12-01 14:35:00 UTC	0.0	nafl	United Kingdom
10	536589	21777	nafl	-10	2010-12-01 16:30:00 UTC	0.0	nafl	United Kingdom

페이지당 결과 수501 - 10 (전체 10건)

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
select count(*)  
from oceanic-citadel-439400-c9.molab.data  
  
or  
  
select *  
from oceanic-citadel-439400-c9.molab.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 invoiceno  
    , count(stockcode) as stock  
    , count(description) as descript  
    , count(quantity) as quantity  
    , count(invoicedate) as invoicedate  
    , count(unitprice) as price  
    , count(customerid) as c_id  
    , count(country) as country  
from oceanic-citadel-439400-c9.molab.data
```

[결과 이미지를 넣어주세요]

행	invoiceno	stock	descript	quantity	invoicedate	price	c_id	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 max(if(abc = 'invoiceno', null_count, 0)) as invoiceno
      , max(if(abc = 'stockcode', null_count, 0)) as stockcode
      , max(if(abc = 'description', null_count, 0)) as description
      , max(if(abc = 'quantity', null_count, 0)) as quantity
      , max(if(abc = 'invoicedate', null_count, 0)) as invoicedate
      , max(if(abc = 'unitprice', null_count, 0)) as unitprice
      , max(if(abc = 'customerid', null_count, 0)) as customerid
      , max(if(abc = 'country', null_count, 0)) as country
from (
  select 'invoiceno' as abc
        , count(*) as null_count
  from oceanic-citadel-439400-c9.molab.data
  where invoiceno is null

  union all

  select 'stockcode' as abc
        , count(*) as null_count
  from oceanic-citadel-439400-c9.molab.data
  where stockcode is null

  union all

  select 'description' as abc
        , count(*) as null_count
  from oceanic-citadel-439400-c9.molab.data
  where description is null

  union all

  select 'quantity' as abc
        , count(*) as null_count
  from oceanic-citadel-439400-c9.molab.data
  where quantity is null

  union all

  select 'invoicedate' as abc
        , count(*) as null_count
  from oceanic-citadel-439400-c9.molab.data
  where invoicedate is null

  union all

  select 'unitprice' as abc
        , count(*) as null_count
  from oceanic-citadel-439400-c9.molab.data
```

```

where unitprice is null

union all

select 'customerid' as abc
      , count(*) as null_count
from oceanic-citadel-439400-c9.molab.data
where customerid is null

union all

select 'country' as abc
      , count(*) as null_count
from oceanic-citadel-439400-c9.molab.data
where country is null
)

```

[결과 이미지를 넣어주세요]

행	invoiceno	stockcode	description	quantity	invoicedate	unitprice	customerid	country
1	0	0	1454	0	0	0	135080	0

## 결측치 처리 전략

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

```

select distinct description
from oceanic-citadel-439400-c9.molab.data
where 1=1
and stockcode = '85123A'

```

[결과 이미지를 넣어주세요]

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

## 결측치 처리

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

```

DELETE FROM oceanic-citadel-439400-c9.molab.data
WHERE 1=1
and (customerid is null
or
description is null)

```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
<p><b>i</b> 이 문으로 data의 행 135,080개가 삭제되었습니다.</p>			

## 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 cnt_dup
from (
  select invoiceno
    , stockcode
    , description
    , quantity
    , invoicedate
    , unitprice
    , customerid
    , country
    , count(*) as cnt
  from oceanic-citadel-439400-c9.molab.data
  group by
    invoiceno
    , stockcode
    , description
    , quantity
    , invoicedate
    , unitprice
    , customerid
    , country
  having count(*) > 1
)
```

[결과 이미지를 넣어주세요]

행	cnt_dup ▼
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 oceanic-citadel-439400-c9.molab.data as
select distinct *
from oceanic-citadel-439400-c9.molab.data
```

[결과 이미지를 넣어주세요]

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

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
select count(distinct invoiceno) as cnt_invoice
from oceanic-citadel-439400-c9.molab.data
```

[결과 이미지를 넣어주세요]

행	cnt_invoice
1	22190

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

```
select distinct invoiceno
from oceanic-citadel-439400-c9.molab.data
limit 100
```

[결과 이미지를 넣어주세요]

행	invoiceno
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032
7	561387
8	574060

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

```
select *
from oceanic-citadel-439400-c9.molab.data
where 1=1
and regexp_contains(invoiceno, '^C')
limit 100
```

[결과 이미지를 넣어주세요]

행	invoiceno	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C575531	22940	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
2	C556080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
3	C556080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
4	C556963	47916A	BLEU HAPPY BIRTHDAY BUNTL.	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554883	475900	PINK HAPPY BIRTHDAY BUNTL.	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
6	C591709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
7	C591709	84978	HANGING HEART JAR TULIGHT	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom

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

```
select round(sum(case when regexp_contains(invoiceno, '^C') then 1 else 0 end) / count(*) * 100, 1) || '%'
from oceanic-citadel-439400-c9.molab.data
```

[결과 이미지를 넣어주세요]

행	per_cnl
1	2.2%

## StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
select count(distinct stockcode)
from oceanic-citadel-439400-c9.molab.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 cnt_sell
from oceanic-citadel-439400-c9.molab.data
group by 1
order by 2 desc
limit 10
```

[결과 이미지를 넣어주세요]

행	stockcode ▼	cnt_sell ▼
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 oceanic-citadel-439400-c9.molab.data
)
where 1=1
and number_count between 0 and 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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
with total as (
  select count(*) as total
  from oceanic-citadel-439400-c9.molab.data
),

filtered as (
  select distinct stockcode
  from (
    select stockcode,
      length(stockcode) - length(regexp_replace(stockcode, r'[0-9]', '')) as number_count
    from oceanic-citadel-439400-c9.molab.data
  )
  where number_count between 0 and 1
),

filtered_cnt as (
  select count(*) as filtered_total
  from oceanic-citadel-439400-c9.molab.data
  where stockcode in (select stockcode from filtered)
)

select round(filtered_cnt.filtered_total / total.total * 100, 2) || '%' as percentage
from filtered_cnt, total;
```

[결과 이미지를 넣어주세요]

행	percentage
1	0.48%

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM oceanic-citadel-439400-c9.molab.data
WHERE StockCode IN (
  select distinct stockcode
  from (
    select stockcode,
      length(stockcode) - length(regexp_replace(stockcode, r'[0-9]', '')) as number_count
    from oceanic-citadel-439400-c9.molab.data
  )
  where number_count between 0 and 1
)
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
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 oceanic-citadel-439400-c9.molab.data
group by 1
order by 2 desc
limit 30
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	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	
7	SET OF 3 CAKE TINS PANTRY ...		1224	
8	LUNCH BAG BLACK SKULL.		1099	
9	PACK OF 72 RETROSPOT CAKE...		1062	
10	SPOTTY BUNTING		1026	
11	PAPER CHAIN KIT 50'S CHRIST...		1013	

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM oceanic-citadel-439400-c9.molab.data
WHERE regexp_contains(description, 'Next Day Carriage|High Resolution Image')
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 data의 행 83개가 삭제되었습니다.			

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE oceanic-citadel-439400-c9.molab.data AS
SELECT
  * EXCEPT (Description),
  upper(description) as Description
FROM oceanic-citadel-439400-c9.molab.data
```



[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
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 oceanic-citadel-439400-c9.molab.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 min(quantity) AS min_quantity
      , max(quantity) AS max_quantity
      , avg(quantity) AS avg_quantity
FROM oceanic-citadel-439400-c9.molab.data
where unitprice = 0
```

[결과 이미지를 넣어주세요]

행	min_quantity	max_quantity	avg_quantity
1	1	12540	420.5151515151...

- UnitPrice = 0를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE oceanic-citadel-439400-c9.molab.data AS
SELECT *
FROM oceanic-citadel-439400-c9.molab.data
WHERE unitprice != 0
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 data인 테이블이 교체되었습니다.			

## 11-7. RFM 스코어

### Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT format_date('%Y-%m-%d', invoicedate) AS InvoiceDay, *
FROM oceanic-citadel-439400-c9.molab.data
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceID	ProductID	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2010-12-21	539722	23403	10	2010-12-21 13:45:00 UTC	0.0	14911	ERE	REGENCY CASESTAND 3 T
2	2011-08-11	502973	23107	240	2011-08-11 11:42:00 UTC	0.0	14911	ERE	SET OF 6 NATIVITY MAGN
3	2011-07-28	501665	23960	11	2011-07-28 17:29:00 UTC	0.0	12007	Spain	JARA MAGNIFIC SET WITH A
4	2011-11-18	577514	23407	2	2011-11-18 12:33:00 UTC	0.0	12644	Norway	SET OF 2 STAFF HOME BR
5	2010-12-05	537197	23841	1	2010-12-05 14:02:00 UTC	0.0	12647	Germany	ROUND CASE TIN VINTAGE
6	2011-11-04	574869	23285	12	2011-11-04 11:35:00 UTC	0.0	12431	Australia	JANARD MAG SPACEBOY DE
7	2011-02-25	554937	23219	80	2011-02-25 14:13:00 UTC	0.0	12415	Australia	SET OF 6 SOLID BRITTEL
8	2011-11-03	574138	23204	216	2011-11-03 11:26:00 UTC	0.0	12415	Australia	BISCUIT TIN VINTAGE CHR
9	2011-08-26	564651	23270	96	2011-08-26 14:19:00 UTC	0.0	14660	Netherlands	SET OF 2 CERAMIC FRANT
10	2011-08-26	564651	23205	144	2011-08-26 14:19:00 UTC	0.0	14660	Netherlands	36 POL STAR CASE CASE

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT max(invoicedate) AS most_recent_date
FROM oceanic-citadel-439400-c9.molab.data
```

[결과 이미지를 넣어주세요]

행	most_recent_date
1	2011-12-09 12:50:00 UTC

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT customerid
      , max(invoicedate) as recency
FROM oceanic-citadel-439400-c9.molab.data
group by 1
```

[결과 이미지를 넣어주세요]

행	customerid	recency
1	12544	2011-11-10 11:12:00 UTC
2	13568	2011-06-19 14:42:00 UTC
3	13824	2011-11-07 12:41:00 UTC
4	14080	2011-11-07 11:09:00 UTC
5	14336	2011-11-23 11:40:00 UTC
6	14592	2011-11-04 16:35:00 UTC
7	15104	2011-06-26 11:35:00 UTC
8	15360	2011-10-31 09:35:00 UTC
9	15872	2011-11-25 11:55:00 UTC

- 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 oceanic-citadel-439400-c9.molab.data
  GROUP BY CustomerID
)
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	17931	134
2	14861	52
3	18189	77
4	17422	16
5	14105	85
6	13859	326
7	13866	74
8	15658	190
9	13630	5

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

```
CREATE OR REPLACE TABLE oceanic-citadel-439400-c9.molab.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(InvoiceDate) AS InvoiceDay
  FROM oceanic-citadel-439400-c9.molab.data
  GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

작업 정보	<b>결과</b>	실행 세부정보	실행 그래프
<p><b>i</b> 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.</p>			

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT CustomerID
      , count(distinct invoiceno) AS purchase_cnt
FROM oceanic-citadel-439400-c9.molab.data
group by 1
```

[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
9	15872	2

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT customerid
      , sum(quantity) as sum_quantity
FROM oceanic-citadel-439400-c9.molab.data
group by 1
```

[결과 이미지를 넣어주세요]

행	customerid	sum_quantity
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	187

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


```
CREATE OR REPLACE TABLE oceanic-citadel-439400-c9.molab.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT customerid
    , count(distinct invoiceno) as purchase_cnt
  FROM oceanic-citadel-439400-c9.molab.data
  group by 1
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT customerid
    , sum(quantity) as item_cnt
  FROM oceanic-citadel-439400-c9.molab.data
  group by 1
)

-- 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 oceanic-citadel-439400-c9.molab.user_r AS ur
  ON pc.CustomerID = ur.CustomerID
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.			

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
select customerid
  , round(sum(quantity * unitprice), 1) as user_total
```

```
from oceanic-citadel-439400-c9.molab.data
group by customerid
```

[결과 이미지를 넣어주세요]

행	customerid	user_total
1	12544	299.7
2	13568	187.1
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	316.2
10	16128	1880.2

#### ○ 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 oceanic-citadel-439400-c9.molab.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, 0) as user_average
from oceanic-citadel-439400-c9.molab.user_rf rf
left join (
  -- 고객 별 총 지출액
  select customerid
        , round(sum(quantity * unitprice), 0) as user_total
  from oceanic-citadel-439400-c9.molab.data2
  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 oceanic-citadel-439400-c9.molab.user_rfm
```

[결과 이미지를 넣어주세요]

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프	
행	customerid	purchase_cnt	item_cnt	recency	user_total	user_average	
1	12713	1	505	0	794.5	794.5	
2	12792	1	215	256	344.5	344.5	
3	18010	1	60	256	174.8	174.8	
4	15083	1	38	256	88.2	88.2	
5	14569	1	79	1	227.4	227.4	

## 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 oceanic-citadel-439400-c9.molab.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM oceanic-citadel-439400-c9.molab.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM oceanic-citadel-439400-c9.molab.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> <div></div> <div>이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.</div> </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 oceanic-citadel-439400-c9.molab.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
    FROM
      oceanic-citadel-439400-c9.molab.data
```

```

        WHERE CustomerID IS NOT NULL
    )
    GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM oceanic-citadel-439400-c9.molab.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID

```

[결과 이미지를 넣어주세요]

**i** 이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

### 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 oceanic-citadel-439400-c9.molab.user_data AS

WITH TransactionInfo AS (
    SELECT CustomerID
           , count(*) as total_transactions
           , sum(case when regexp_contains(invoiceno, '^C') then 1 else 0 end) as cancel_frequency
    FROM oceanic-citadel-439400-c9.molab.data
    group by 1
)

SELECT u.*
       , t.* EXCEPT(CustomerID)
       , round(t.cancel_frequency / t.total_transactions * 100, 2) as cancel_rate
FROM `oceanic-citadel-439400-c9.molab.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.customerid = t.customerid

```

[결과 이미지를 넣어주세요]

작업 정보   결과   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

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

```

-- RFM 등급별 요약--
with a as (
    select customerid
           , recency
           , purchase_cnt as frequency
           , user_total as monetary
           , ntile(5) over (order by recency) as r_rank

```

```

        , ntile(5) over (order by purchase_cnt desc) as f_rank
        , ntile(5) over (order by user_total desc) as m_rank
    from `oceanic-citadel-439400-c9.molab.user_data`
),
b as (
    select customerid
        , r_rank
        , f_rank
        , m_rank
        , cast(r_rank as string) || '-' || cast(f_rank as string) || '-' || cast(m_rank as string) as rfm_
        , frequency
        , monetary
        , recency
    from a
),
rfm_summary as (
    select r_rank
        , f_rank
        , m_rank
        , count(*) as customer_count
        , avg(frequency) as avg_pur_frequency
        , avg(monetary) as avg_pur_money
        , avg(recency) as avg_recency
    from b
    group by 1, 2, 3
)
select 'Recency Rank' as category
    , r_rank as rank
    , count(*) as total_customers
    , round(avg(avg_pur_frequency), 0) as avg_frequency
    , round(avg(avg_pur_money), 0) as avg_monetary
    , round(avg(avg_recency), 0) as avg_recency
from rfm_summary
group by 2

union all

select 'Frequency Rank' as category
    , f_rank as rank
    , count(*) as total_customers
    , round(avg(avg_pur_frequency), 0) as avg_frequency
    , round(avg(avg_pur_money), 0) as avg_monetary
    , round(avg(avg_recency), 0) as avg_recency
from rfm_summary
group by 2

union all

select 'Monetary Rank' as category
    , m_rank as rank
    , count(*) as total_customers
    , round(avg(avg_pur_frequency), 0) as avg_frequency
    , round(avg(avg_pur_money), 0) as avg_monetary
    , round(avg(avg_recency), 0) as avg_recency
from rfm_summary
group by 2
order by 1, 2

```



[결과 이미지를 넣어주세요]

행	category ▼	rank ▼	total_customers ▼	avg_frequency ▼	avg_monetary ▼	avg_recency ▼
1	Frequency Rank	1	20	10.0	2115.0	88.0
2	Frequency Rank	2	25	5.0	1584.0	84.0
3	Frequency Rank	3	25	3.0	1155.0	87.0
4	Frequency Rank	4	25	2.0	976.0	91.0
5	Frequency Rank	5	23	1.0	1140.0	99.0
6	Monetary Rank	1	24	5.0	4292.0	87.0
7	Monetary Rank	2	24	4.0	1281.0	88.0
8	Monetary Rank	3	25	4.0	656.0	87.0
9	Monetary Rank	4	23	3.0	352.0	98.0
10	Monetary Rank	5	22	3.0	142.0	90.0
11	Recency Rank	1	24	4.0	1531.0	6.0
12	Recency Rank	2	22	4.0	1169.0	21.0
13	Recency Rank	3	23	3.0	1344.0	49.0
14	Recency Rank	4	25	4.0	1178.0	112.0
15	Recency Rank	5	24	4.0	1605.0	253.0

[인사이트]

- Frequency
  - 빈도별로 보면, 빈도가 높을수록 평균 구매 금액이 전반적으로 더 높고, **recency**도 비교적 짧은 경향이 있음. 빈도 1위 고객군은 자주 구매할 뿐만 아니라 구매 금액도 크다는 점에서 가장 중요한 고객군임을 알 수 있음
- Monetary
  - 빈도가 낮아도 구매 금액이 크면 여전히 높은 가치가 있는 고객임. 반대로, 빈도가 높더라도 구매 금액이 낮은 경우는 수익 증대에 좋지 않아 다른 마케팅 제안(?)이 필요해 보임
- Recency
  - 오래 전에 구매한 고객들 중 일부가 높은 구매 금액을 기록했다는 점이 주목할 점. 특히, **Rank 5** 고객들은 최근에 구매하지 않았지만, 이들의 재 참여를 유도(마케팅 제안 필요)는 매출 증대에 큰 중요한 역할을 할 것으로 보임