# 고객을 세그먼테이션하자 [프로젝트]

## 11-2. 데이터 불러오기

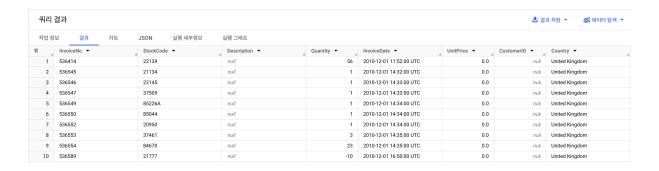
## 데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
-- 11-2-1 간단하게 테이블에 있는 10개의 행만 출력해보겠습니다.
select *
```

from dotted-banner-425501-r3.modulabs\_project.data
limit 10

#### [결과 이미지를 넣어주세요]



#### • 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

-- 11-2-1 전체 데이터는 몇 행으로 구성되어 있는지 확인해 봅시다.

select count(\*) as total\_rows
from dotted-banner-425501-r3.modulabs\_project.data



## 데이터 수 세기

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

```
-- 11-2-2 빅쿼리에서 모든 컬럼에 대하여 COUNT 함수를 적용해 주세요.

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 dotted-banner-425501-r3.modulabs_project.data
```



# 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

```
-- 11-4-2 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNI
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 EN
from dotted-banner-425501-r3.modulabs_project.data
union all
select
    'StockCode' AS column_name,
   ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 EN
from dotted-banner-425501-r3.modulabs_project.data
union all
select
    'Description' AS column_name,
   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0
from dotted-banner-425501-r3.modulabs_project.data
union all
select
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END
from dotted-banner-425501-r3.modulabs project.data
union all
select
```

3

'InvoiceDate' AS InvoiceDate,

ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 from dotted-banner-425501-r3.modulabs\_project.data union all

select

'UnitPrice' AS UnitPrice,

ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 EN from dotted-banner-425501-r3.modulabs\_project.data union all

select

'CustomerID' AS column\_name,

ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 Elfrom dotted-banner-425501-r3.modulabs\_project.data union all

select

'Country' AS column\_name,

ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) from dotted-banner-425501-r3.modulabs\_project.data

[결과 이미지를 넣어주세요]

#### 쿼리 결과 작업 정보 차트 실행 세부 결과 JSON 행 column\_name ▼ missing\_percentage 0.0 1 Country 2 UnitPrice 0.0 3 CustomerID 24.93 InvoiceDate 4 0.0 InvoiceNo 5 0.0 0.27 6 Description StockCode 0.0 7

## 결측치 처리 전략

8

Quantity

• StockCode = '85123A' 의 Description 을 추출하는 쿼리문을 작성하기

```
-- 11-4-4 아래의 결과가 나오도록 StockCode = '85123A'의 Descrip
select distinct Description
from dotted-banner-425501-r3.modulabs_project.data
where StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

0.0

쿼리	결과			
작업 정	보	결과	차트	JSON
행	Desci	ription 🔻		//
1	?			
2	wrong	gly marked o	arton 22804	
3	CREA	M HANGING	HEART T-LIG	GHT HOLDER
4	WHIT	E HANGING	HEART T-LIG	HT HOLDER

## 결측치 처리

- DELETE 구문을 사용하며, WHERE 절을 통해 데이터를 제거할 조건을 제시
  - -- 11-4-5 결측치 제거 delete from dotted-banner-425501-r3.modulabs\_project.data where CustomerID is null
  - -- where 절을 CustomerID is null or StockCode = '85123A'으로
  - -- 삭제 행수가 135,080개 보다 많아집니다.

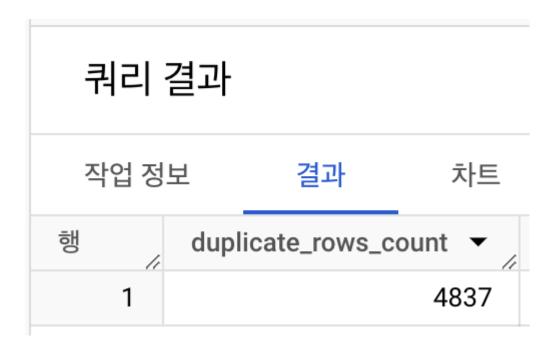
쿼리 결과			
작업 정보	결과	실행 세부정보	실행 그래프
<b>①</b> 이 문의	으로 data3의 <sup>:</sup>	행 135,080개가 삭제되	었습니다.

# 11-5. 데이터 전처리(2): 중복값 처리

### 중복값 확인

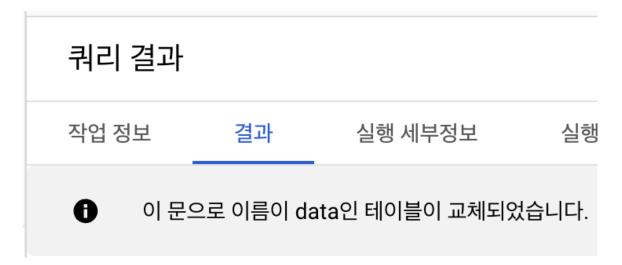
- 중복된 행의 수를 세어보기
  - 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

```
-- 11-5-1 중복된 행의 수를 세어주세요. 8개의 컬럼에 그룹 함수를 적용한 select count(*) as duplicate_rows_count from (
select InvoiceNo, StockCode, Description, Quantity, Inform dotted-banner-425501-r3.modulabs_project.data group by InvoiceNo, StockCode, Description, Quantity, having count(*) > 1
) as duplicates;
```



## 중복값 처리

- 중복값을 제거하는 쿼리문 작성하기
  - CREATE OR REPLACE TABLE 구문을 활용하여 모든 컬럼(\*)을 DISTINCT 한데이터로 업데이트
  - -- 11-5-2 CREATE OR REPLACE TABLE 구문을 활용하여 모든 컬럼(\*)을 create or replace table dotted-banner-425501-r3.modulabs\_p select distinct \* from dotted-banner-425501-r3.modulabs\_project.data
    -- select count(\*)
    -- from dotted-banner-425501-r3.modulabs\_project.data





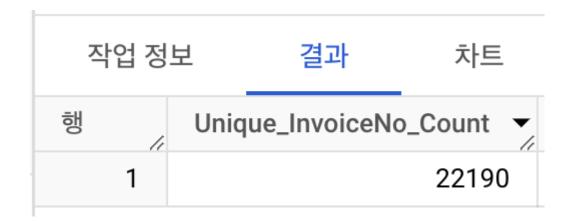
# 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

-- 11-6-1 고유(unique)한 InvoiceNo의 개수를 출력해 보세요.

select count(distinct InvoiceNo) as Unique\_InvoiceNo\_Count from dotted-banner-425501-r3.modulabs\_project.data



#### • 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

-- 11-6-2 이번에는 고유한 InvoiceNo를 앞에서부터 100개를 출력해 보기 select distinct InvoiceNo from dotted-banner-425501-r3.modulabs\_project.data limit 100;

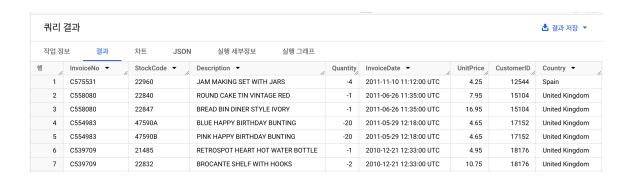
쿼리	결과
작업 정	보 결과
행	InvoiceNo ▼
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032
7	561387
8	574868
9	574827

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

-- 11-6-3 InvoiceNo가 'C'로 시작하는 행을 필터링할 수 있는 쿼리문을

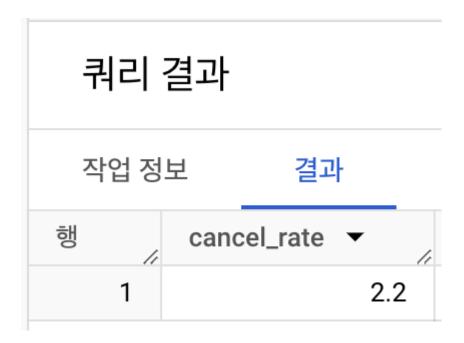
select \*
from dotted-banner-425501-r3.modulabs\_project.data
where InvoiceNo like 'C%'
limit 100;

#### [결과 이미지를 넣어주세요]



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

-- 11-6-4 구매 건 상태가 Canceled 인 데이터의 비율(%)은 어떻게 되니 select round(sum(case when InvoiceNo Like 'C%' then 1 else 0 en as cancel\_rate from dotted-banner-425501-r3.modulabs\_project.data



# StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

-- 11-6-5 우선 고유한 StockCode의 개수를 출력해보겠습니다.

select distinct StockCode
from dotted-banner-425501-r3.modulabs\_project.data

[결과 이미지를 넣어주세요]

쿼리	결과		
작업 정	보 결과	차트	J
행	StockCode ▼		//
1	22751		
2	22144		
3	20749		
4	85049E		
5	23512		
6	23240		
7	22086		
8	22621		
9	85049A		

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기
  - 。 상위 10개의 제품들을 출력하기
    - -- 11-6-6 이번에는 어떤 제품이 가장 많이 판매되었는지 보기 위하여 Stc

```
select StockCode, count(*) as sell_cnt
from dotted-banner-425501-r3.modulabs_project.data
group by 1
order by 2 desc
limit 10;
```

쿼리	결과					
작업 정	보	결과	차트	J	SON	실행 세부
행 //	Stock	Code ▼		li	sell_cnt	<b>▼</b>
1	85123	A		,,,		2065
2	22423					1894
3	85099	В				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

```
-- 11-6-8 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지를 확인한 select distinct StockCode, number_count from (
   select StockCode,
   length(StockCode) - length(regexp_replace(StockCode, r from dotted-banner-425501-r3.modulabs_project.data
)
where number_count in (0,1);
```

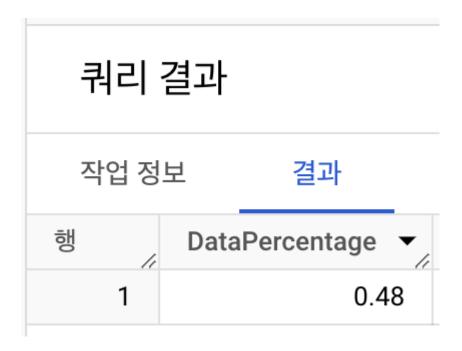
쿼리	결과			
작업 정	보	결과	차트	≣ JSON
행 //	StockCod	e ▼	h	number_count
1	POST			0
2	М			0
3	PADS			0
4	D			0
5	BANK CH	ARGES		0
6	DOT			0
7	CRUK			0
8	C2			1

- 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)

```
-- 11-6-9 숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 with NumberCount as (
    select StockCode,
    length(StockCode) - length(regexp_replace(StockCode, r from dotted-banner-425501-r3.modulabs_project.data),

ZeroOne as (
    select StockCode
    from NumberCount
    where number_count in (0, 1)
)

select
    round((count(*) / (select count(*) from dotted-banner-4256))
as DataPercentage
from ZeroOne;
```

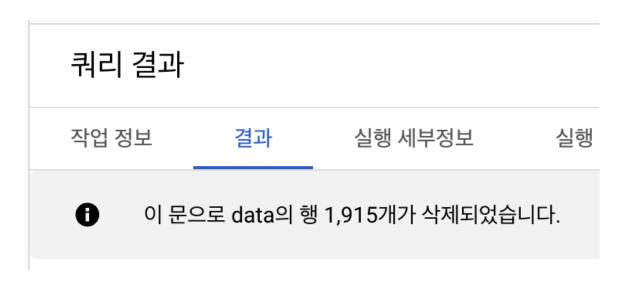


#### • 제품과 관련되지 않은 거래 기록을 제거하기

```
-- 11-6-10 제품과 관련되지 않은 거래 기록을 제거하는 쿼리문을 작성해

delete from dotted-banner-425501-r3.modulabs_project.data
where StockCode in (
    select distinct StockCode
    from (
        select StockCode,
        length(StockCode) - length(regexp_replace(StockCode,
        from dotted-banner-425501-r3.modulabs_project.data
    )
    where number_count in (0,1)
)
```

[결과 이미지를 넣어주세요]



# Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
-- 11-6-11 먼저 데이터셋에서 고유한 Description 별 출현 빈도를 계신 select Description, count(*) as description_cnt from dotted-banner-425501-r3.modulabs_project.data group by 1 order by 2 desc limit 30;
```

[결과 이미지를 넣어주세요]

쿼리 결과					
작업 정	보	결과	차트	JSON	실행 세부정보
행	Descri	iption 🔻		//	description_cnt ▼
1	WHITE	HANGING	HEART T-LIG	HT HOLDER	2058
2	REGEN	NCY CAKES	TAND 3 TIER		1894
3	JUMB	O BAG RED	RETROSPOT		1659
4	PARTY	/ BUNTING			1409
5	ASSOF	RTED COLOU	JR BIRD ORN	AMENT	1405
6	LUNC	H BAG RED I	RETROSPOT		1345
7	SET O	F 3 CAKE TI	NS PANTRY	DESIGN	1224
8	LUNC	HBAG BLAG	CK SKULL.		1099
9	PACK	OF 72 RETR	OSPOT CAKI	E CASES	1062
10	SPOTT	TY BUNTING	i		1026

#### • 서비스 관련 정보를 포함하는 행들을 제거하기

```
-- 11-6-13 우선 서비스 관련 정보를 포함하는 행들을 제거하는 쿼리문을

delete
from dotted-banner-425501-r3.modulabs_project.data
where
   Description like '%Next Day Carriage%'
   or Description like '%High Resolution Image%';
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보	결과	실행 세부정보
<b>&amp;</b> 이 문의	으로 data의 형	행 83개가 삭제되었습니다.

#### • 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

-- 11-6-14 이번에는 대소문자를 혼합하고 있는 데이터를 대문자로 표준화경 create or replace table dotted-banner-425501-r3.modulabs\_p select \* except (Description), upper(Description) as Description from dotted-banner-425501-r3.modulabs\_project.data;

### [결과 이미지를 넣어주세요]

쿼리 결과			
작업 정보	결과	실행 세부정보	실행
<b>6</b> 이 문으	으로 이름이 da	ata인 테이블이 교체되었	습니다.

## UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```
-- 11-6-15 UnitPrice의 최솟값, 최댓값, 평균을 구해 보세요.
select min(UnitPrice) as min_price, max(UnitPrice) as max_
from dotted-banner-425501-r3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



• 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 dotted-banner-425501-r3.modulabs_project.data
where UnitPrice = 0;
```

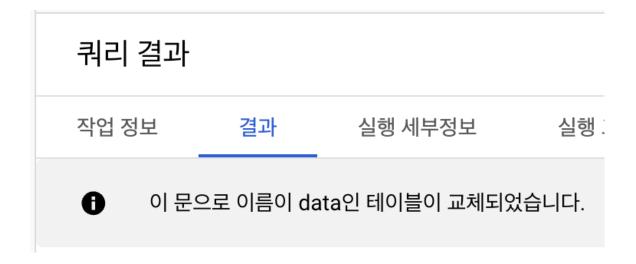
[결과 이미지를 넣어주세요]

쿼리	결과					
작업 정	보	결과	차트	JSON	실행 세부정보	실험
행	cnt_	quantity	min_quantity	max_quantity	avg_quantity 🔻	11
1		33	1	12540	420.5151515151	515

#### • UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
-- 11-6-17 이 데이터(UnitPrice = 0)를 제거하고 일관된 데이터셋을 ና
create or replace table dotted-banner-425501-r3.modulabs_p
select *
from dotted-banner-425501-r3.modulabs_project.data
where UnitPrice != 0;
```

#### [결과 이미지를 넣어주세요]



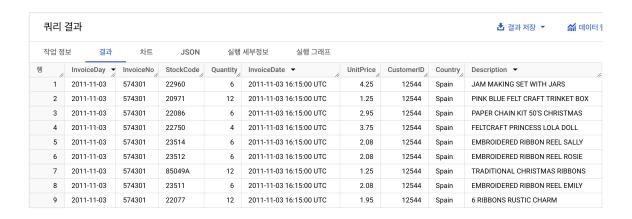
# 11-7. RFM 스코어

### Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

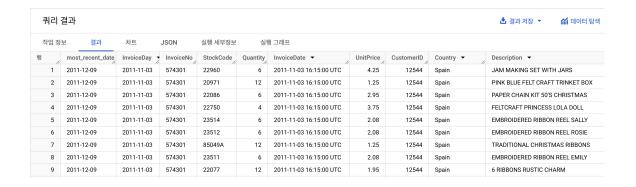
```
-- 11-7-1 DATE 함수를 활용하여 InvoiceDate 컬럼을 연월일 자료형으로
select date(InvoiceDate) as InvoiceDay, *
from dotted-banner-425501-r3.modulabs_project.data
```

#### [결과 이미지를 넣어주세요]



#### • 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
-- 11-7-2 우선 가장 최근 구매 일자를 MAX() 함수로 찾아보겠습니다.
select
  (select
   date(max(InvoiceDate))
   from dotted-banner-425501-r3.modulabs_project.data)
   as most_recent_date,
   date(InvoiceDate) as InvoiceDay,
  *
from dotted-banner-425501-r3.modulabs_project.data
```



#### • 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

-- 11-7-3 이번에는 유저 별로 가장 최근에 일어난 구매 정보를 정리해 봅 select
 CustomerID,
 max(date(InvoiceDate)) as InvoiceDay
from dotted-banner-425501-r3.modulabs\_project.data
Group by 1

[결과 이미지를 넣어주세요]

쿼	리	결과

작업 정	보 결과	차트	JSON
행	CustomerID ▼	InvoiceDay	<b>▼</b>
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	
9	15872	2011-11-25	

가장 최근 일자(most\_recent\_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산하기

```
-- 11-7-4 다음에는 가장 최근 일자(most_recent_date)와 유저별 마지
SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) A
FROM (
SELECT
```

```
CustomerID,

MAX(DATE(InvoiceDate)) AS InvoiceDay

from dotted-banner-425501-r3.modulabs_project.data

GROUP BY CustomerID
);
```

쿼	己	결과
- 17	_	

작업 정	보 결과	차트	JSON
행	CustomerID ▼	recency	<b>▼</b>
1	17171		289
2	15657		22
3	14894		85
4	13103		39
5	13359		59
6	12599		29
7	12348		75
8	14141		2
9	16959		87

• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user\_r 이라는 이름의 테이블로 저장하기

```
-- 17-7-5 이제 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙 create or replace table dotted-banner-425501-r3.modulabs_p select
   CustomerID,
   extract(day from most_recent_date - InvoiceDay) as recenfrom (
   select
        CustomerID,
        max(date(InvoiceDate)) as InvoiceDay,
        (select
              date(max(InvoiceDate))
        from dotted-banner-425501-r3.modulabs_project.data) a
   from dotted-banner-425501-r3.modulabs_project.data
   group by 1
);
```

[결과 이미지를 넣어주세요]

# ■ user\_r





스키마	세부정보	미리보기
행	CustomerID	recency
1	14422	0
2	12518	0
3	13426	0
4	17754	0
5	15910	0
6	12985	0
7	15344	0
8	16446	0
9	14441	0
10	12423	0
11	13777	0
12	16558	0

## **Frequency**

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
-- 17-7-6 우선 각 고객의 거래 건수를 세어 봅시다. 거래 건은 Invoice select
   CustomerID,
   count(distinct InvoiceNo) as purchase_cnt
from dotted-banner-425501-r3.modulabs_project.data
group by 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과								
보 결과	차트 JSON							
CustomerID ▼	purchase_cnt ▼							
12544	2							
13568	1							
13824	5							
14080	1							
14336	4							
14592	3							
15104	3							
15360	1							
15872	2							
16128	5							
	보 결과  CustomerID ▼  12544  13568  13824  14080  14336  14592  15104  15360  15872							

### • 각 고객 별로 구매한 아이템의 총 수량 더하기

-- 11-7-7 그 다음으로는 각 고객 별로 구매한 아이템의 총 수량을 더해주 select

CustomerID,

sum(Quantity) as item\_cnt

from dotted-banner-425501-r3.modulabs\_project.data
group by 1;

[결과 이미지를 넣어주세요]

쿼리 결과								
작업 정	작업 정보 <mark>결과</mark> 차트 JSO							
행 //	CustomerID ▼	item_cnt	<b>▼</b>					
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 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 user\_rf 라는 이름의 테이블에 저장하기

-- 11-7-8 이제 위에서 구한 '1. 전체 거래 건수 계산'과 '2. 구매한 아 create or replace table dotted-banner-425501-r3.modulabs\_p

```
--(1) 전체 거래 건수 계산
with purchase_cnt as (
 select
   CustomerID,
   count(distinct InvoiceNo) as purchase_cnt
 from dotted-banner-425501-r3.modulabs_project.data
 group by 1
),
--(2) 구매한 아이템 총 수량 계산
item_cnt as (
  select
   CustomerID,
    sum(Quantity) as item_cnt
 from dotted-banner-425501-r3.modulabs_project.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 dotted-banner-425501-r3.modulabs_project.user_r as ur
on pc.CustomerID = ur.CustomerID;
```

<b>⊞</b> us	er_rf	Q 쿼리 ▼ +9	공유 □복사	₫ 스냅샷
스키마	세부정보	미리보기	계보	데이터 프로필
행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	18010	1	60	256
3	15083	1	38	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

## **Monetary**

#### • 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
-- 11-7-9 고객별 총 지출액을 계산해 보세요. 소수점 첫째 자리에서 반을
select
CustomerID,
round(sum(Quantity * UnitPrice), 1) as user_total
from dotted-banner-425501-r3.modulabs_project.data
group by 1
```

# 쿼리 결과

작업 정	보 결과	차트	JSON
행 //	CustomerID ▼	user_total	<b>▼</b>
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		316.3
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

```
-- 11-7-10 고객별 평균 거래 금액을 구하기 위해 1) data 테이블을 use
create or replace table dotted-banner-425501-r3.modulabs_p
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 dotted-banner-425501-r3.modulabs_project.user_rf rf
left join (
  -- 고객 별 총 지출액
 select
   CustomerID,
    round(sum(Quantity * UnitPrice), 1) as user_total
 from dotted-banner-425501-r3.modulabs_project.data
 group by 1
) ut
on rf.CustomerID = ut.CustomerID;
```

<b>⊞</b> us	er_rfm	Q 쿼리 ▼	<b>*</b>	복사 🛨 스냅샷	출 삭제	<b>⚠</b> 내보내기 ▼
스키마	세부정보	미리보기	계보	데이터 프로필	데이터 품질	
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.6	794.6
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	14569	1	79	1	227.4	227.4
7	13298	1	96	1	360.0	360.0
8	13436	1	76	1	196.9	196.9
9	13357	1	321	257	609.4	609.4
10	14476	1	110	257	193.0	193.0

## RFM 통합 테이블 출력하기

### • 최종 user\_rfm 테이블을 출력하기

-- 11-7-11 위와 같은 과정을 거쳐서 생성된 최종 user\_rfm 테이블을 출 select \*

from dotted-banner-425501-r3.modulabs\_project.user\_rfm

#### [결과 이미지를 넣어주세요]

쿼리	결과					盘 결과 저		
작업 정	작업 정보 결과 차트 JSON 실행 세부정보 설							
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average		
1	12713	1	505	0	794.6	794.6		
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	14569	1	79	1	227.4	227.4		
7	13298	1	96	1	360.0	360.0		
8	13436	1	76	1	196.9	196.9		
9	13357	1	321	257	609.4	609.4		
10	14476	1	110	257	193.0	193.0		

# 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

```
-- 11-8-1 우선 1) 고객 별로 구매한 상품들의 고유한 수를 계산합니다.
-- 높은 숫자가 나오는 것은 해당 고객이 다양한 제품들을 구매한다는 의미이며,
-- 낮은 값이 나오는 경우 소수의 제품들만 구매한다는 것을 의미합니다.
-- 이후 2) user rfm 테이블과 결과를 합치고, 이를 3) user data라는 이탈
CREATE OR REPLACE TABLE dotted-banner-425501-r3.modulabs proje
WITH unique_products AS (
 SELECT
   CustomerID,
   COUNT(DISTINCT StockCode) AS unique_products
 FROM dotted-banner-425501-r3.modulabs_project.data
 GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM dotted-banner-425501-r3.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

#### [결과 이미지를 넣어주세요]

<b>⊞</b> u	ser_data	<b>Q</b> 쿼리 ▼	<b>+_</b> 공유 <b>[</b>	]복사 🔳 스냅/	샷 📋 삭제	▲ 내보내기 ▼		€새로고
스키디	마 세부정보	미리보기	계보	데이터 프로필	데이터 품질			
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	14432	6	2013	9	2248.5	374.8	256	0.2
2	12428	11	3477	25	6366.0	578.7	256	0.87
3	13268	14	3525	17	3105.7	221.8	256	0.56
4	15118	1	1440	134	244.8	244.8	1	0.0
5	13135	1	4300	196	3096.0	3096.0	1	0.0
6	13270	1	200	366	590.0	590.0	1	0.0
7	16990	1	100	218	179.0	179.0	1	0.0
8	13188	1	24	11	99.6	99.6	1	0.0
9	16138	1	-1	368	-8.0	-8.0	1	0.0
10	18133	1	1350	212	931.5	931.5	1	0.0
11	14576	1	12	372	35.4	35.4	1	0.0

### 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

```
-- 11-8-2 평균 구매 소요 일수를 계산하고, 그 결과를 user_data에 통합해
CREATE OR REPLACE TABLE dotted-banner-425501-r3.modulabs proje
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 RO
 FROM (
   -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION
   FROM
     dotted-banner-425501-r3.modulabs_project.data
   WHERE CustomerID IS NOT NULL
 GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM dotted-banner-425501-r3.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

<b>⊞</b> us	er_data	<b>Q</b> 쿼리 ▼	<b>*</b> 로공유 <b>[</b>	복사 🛨 스냅:	샷 📋 삭제	<b>⚠</b> 내보내기 ▼	
스키마	세부정보	미리보기	계보	데이터 프로필	데이터 품질		
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	12603	1	56	21	613.2	613.2	1
2	16344	1	18	158	101.1	101.1	1
3	14351	1	12	164	51.0	51.0	1
4	16765	1	4	294	34.0	34.0	1
5	16454	1	2	64	5.9	5.9	1
6	17347	1	216	86	229.0	229.0	1
7	14705	1	100	198	179.0	179.0	1
8	16148	1	72	296	76.3	76.3	1
9	16737	1	288	53	417.6	417.6	1
10	16138	1	-1	368	-8.0	-8.0	1
11	17331	1	16	123	175.2	175.2	1

## 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)

```
-- 11-8-3 취소 빈도와 취소 비율을 계산하고 그 결과를 user_data에 통합해 create or replace table dotted-banner-425501-r3.modulabs_projewith TransactionInfo as (
    select
        CustomerID,
        count(InvoiceNo) as total_transactions,
        sum(case when left(InvoiceNo, 1) = 'C' then 1 else 0 end)
    from dotted-banner-425501-r3.modulabs_project.data
    group by 1
)

select u.*, t.* except(CustomerID),
```

round(ifnull(t.cancel\_frequency / nullif(t.total\_transactio
from dotted-banner-425501-r3.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

-- 11-8-4 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적 select \*

from dotted-banner-425501-r3.modulabs\_project.user\_data

