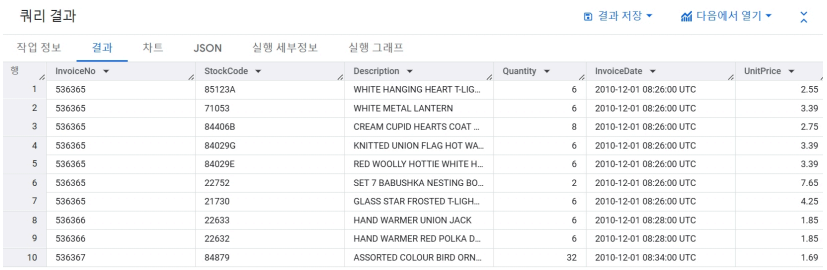
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
select *
from `sincere-burner-456110-d0.modulabs_project.data`
limit 10
```



쿼리 결과

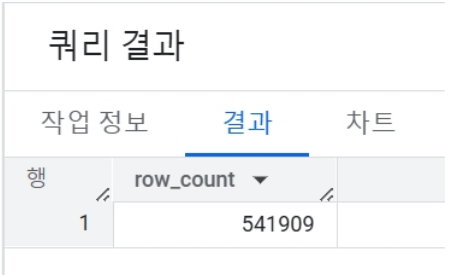
결과 저장 ▾ 다음에서 열기 ▾

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

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

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
select count(*) as row_count
from sincere-burner-456110-d0.modulabs_project.data
```



쿼리 결과

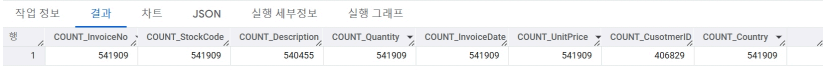
작업 정보 | 결과 | 차트

행	row_count
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_CusotmerID
      , count(Country) as COUNT_Country
from sincere-burner-456110-d0.modulabs_project.data
```



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

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CusotmerID	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**

```
-- 데이터 전처리(1): 결측치 제거

select 'InvoiceNo' as column_name,
       round(sum(case when InvoiceNo is null then 1 else 0 end) / count(*) * 100, 2) as missing_percentage
from sincere-burner-456110-d0.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) as missing_percentage
from sincere-burner-456110-d0.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 sincere-burner-456110-d0.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 missing_percentage
from sincere-burner-456110-d0.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 sincere-burner-456110-d0.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 missing_percentage
from sincere-burner-456110-d0.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 sincere-burner-456110-d0.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 missing_percentage
from sincere-burner-456110-d0.modulabs_project.data
```

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

결측치 처리 전략

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

```
select distinct Description
from sincere-burner-456110-d0.modulabs_project.data
where StockCode = '85123A'
order by Description asc
```

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

결측치 처리

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

```
-- 데이터 전처리(1): 결측치 처리(제거)
delete from sincere-burner-456110-d0.modulabs_project.data
where InvoiceNo is null or
    StockCode is null or
    Description is null or
    Quantity is null or
    InvoiceDate is null or
    UnitPrice is null or
    CustomerID is null or
    Country is null
```

이 문으로 data의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

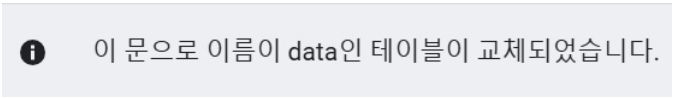
```
-- 데이터 전처리(2): 중복값 확인
select count(*) as duplicate_row_count
from (select *, count(*) as cnt
    from sincere-burner-456110-d0.modulabs_project.data
    group by all
    having count(*) > 1);
```

행	duplicate_row_count
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 sincere-burner-456110-d0.modulabs_project.data as
select distinct *
from sincere-burner-456110-d0.modulabs_project.data
```

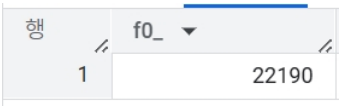


11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```
select count(distinct InvoiceNo)
from sincere-burner-456110-d0.modulabs_project.data
```



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

```
select distinct InvoiceNo
from sincere-burner-456110-d0.modulabs_project.data
limit 100
```

행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318
11	541998
12	548955
13	568172
14	577609

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

```
select *
from sincere-burner-456110-d0.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
limit 100
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDa
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-1
2	C545329	M	Manual	-1	2011-03-(
3	C545329	M	Manual	-1	2011-03-(
4	C545330	M	Manual	-1	2011-03-(
5	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03-2
6	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-2
7	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-2
8	C547388	22701	PINK DOG BOWL	-6	2011-03-2
9	C547388	84050	PINK HEART SHAPE EGG FRYI...	-12	2011-03-2
10	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-2
11	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12	2011-03-2
12	C549955	22839	3 TIER CAKE TIN GREEN AND ...	-2	2011-04-1

- 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 sincere-burner-456110-d0.modulabs_project.data
```

f0_
2.2

`StockCode` 살펴보기

- 고유한 `StockCode`의 개수를 출력하기

```
select count(distinct StockCode) as StockCode_counts
```

```
from sincere-burner-456110-d0.modulabs_project.data
```

행	StockCode_counts
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 sincere-burner-456110-d0.modulabs_project.data
group by StockCode
order by sell_cnt desc
limit 10
```

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 sincere-burner-456110-d0.modulabs_project.data )
where number_count between 0 and 1
```

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)

```
WITH processed AS (
  SELECT
    StockCode,
    REGEXP_REPLACE(StockCode, r'[0-9]', '') AS only_letters,
    LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS letter_count,
    ARRAY_LENGTH(REGEXP_EXTRACT_ALL(StockCode, r'[0-9]')) AS digit_count
  FROM
    sincere-burner-456110-d0.modulabs_project.data
),
filtered AS (
  SELECT *
  FROM processed
  WHERE digit_count <= 1
)
SELECT
  ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) AS total_count FROM sincere-burner-456110-d0.modulabs_project.data), 2) A
FROM filtered
```

행	percentage
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
delete from sincere-burner-456110-d0.modulabs_project.data
where StockCode in (select distinct StockCode
                    from (select * from (SELECT StockCode,
                                          REGEXP_REPLACE(StockCode, r'[0-9]', '') AS only_letters,
                                          LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS letter_count,
                                          ARRAY_LENGTH(REGEXP_EXTRACT_ALL(StockCode, r'[0-9]')) AS digit_count
                                        FROM sincere-burner-456110-d0.modulabs_project.data ) where digit_count <= 1))
```

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 sincere-burner-456110-d0.modulabs_project.data
group by Description
```

행	Description	description_cnt
1	MEDIUM CERAMIC TOP STORA...	208
2	AIRLINE BAG VINTAGE JET SE...	96
3	CAMOUFLAGE EAR MUFF HEA...	17
4	ALARM CLOCK BAKELIKE PINK	636
5	RED DRAWER KNOB ACRYLIC ...	101
6	BLACK EAR MUFF HEADPHON...	18
7	BLACK GRAND BAROQUE PHO...	7
8	PURPLE DRAWERKNOB ACRYL...	151
9	COLOUR GLASS. STAR T-LIGHT...	247
10	ALARM CLOCK BAKELIKE ORA...	382
11	BOX OF 6 ASSORTED COLOUR ...	75
12	BLACK CANDELABRA T-LIGHT ...	40
13	ALARM CLOCK BAKELIKE GRE...	819

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
-- 서비스 관련 정보를 포함하는 행들을 제거하기
delete from sincere-burner-456110-d0.modulabs_project.data
where REGEXP_CONTAINS(Description, r'\b(cm|g|Next Day Carriage|High Resolution Image)\b')
```

이 문으로 data의 행 83개가 삭제되었습니다.

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
create or replace table `sincere-burner-456110-d0.modulabs_project.data` as
select * except(Description),
      upper(Description) as Description
from sincere-burner-456110-d0.modulabs_project.data
```

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 sincere-burner-456110-d0.modulabs_project.data
```

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**

```
-- 단가가 0원인 거래의 개수, 구매수량(Quantity)의 최소값, 최대값, 평균 구하기
select count(Quantity) as cnt_quantity, min(Quantity) as min_quantity, max(Quantity) as max_quantity, avg(Quantity) as avg_qua
from sincere-burner-456110-d0.modulabs_project.data
where UnitPrice = 0.0
```

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 sincere-burner-456110-d0.modulabs_project.data as
select *
from sincere-burner-456110-d0.modulabs_project.data
where not (UnitPrice = 0.0)
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

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

```
select date(InvoiceDate) as InvoiceDay
from sincere-burner-456110-d0.modulabs_project.data
```

행	InvoiceDay
1	2011-01-18
2	2011-01-18
3	2010-12-07
4	2010-12-07
5	2010-12-07
6	2010-12-07
7	2010-12-07
8	2010-12-07
9	2010-12-07
10	2010-12-07
11	2010-12-07
12	2010-12-07
13	2010-12-07
14	2010-12-07
15	2010-12-07
16	2010-12-07

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
select max(date(InvoiceDate)) as most_recent_day
from sincere-burner-456110-d0.modulabs_project.data
```

행	most_recent_day
1	2011-12-09

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

```
select CustomerID, max(date(InvoiceDate)) as InvoiceDay
from sincere-burner-456110-d0.modulabs_project.data
group by CustomerID
```

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
10	12356	2011-11-17
11	12357	2011-11-06
12	12358	2011-12-08
13	12359	2011-12-02
14	12360	2011-10-18
15	12361	2011-02-25
16	12362	2011-12-06

- **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
);
```

행	CustomerID	recency
1	12609	78
2	12716	3
3	12744	56
4	12906	11
5	12956	306
6	13130	94
7	13261	268
8	13270	366
9	13599	1
10	13947	65
11	13950	11
12	14227	26
13	14272	73
14	14314	106
15	14360	31
16	14592	35

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

```
CREATE OR REPLACE TABLE sincere-burner-456110-d0.modulabs_project.user_r
(
    CustomerID int64,
    recency int64
);

insert into sincere-burner-456110-d0.modulabs_project.user_r(
    CustomerID, recency) (select CustomerID, extract(day from max(InvoiceDay) over() - InvoiceDay) as recency
                    from (
                        select CustomerID, max(date(InvoiceDate)) as InvoiceDay
                        from sincere-burner-456110-d0.modulabs_project.data
                        group by CustomerID)
    )
```

행	CustomerID	recency
1	17001	0
2	16446	0
3	17490	0
4	14397	0
5	14441	0
6	15311	0
7	12680	0
8	16558	0
9	12748	0
10	17581	0
11	14422	0
12	12518	0
13	13069	0
14	12713	0
15	12985	0
16	17754	0
17	17428	0
18	13777	0
19	15910	0
20	16626	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
select CustomerID, count(distinct InvoiceNo) as purchase_cnt
from sincere-burner-456110-d0.modulabs_project.data
group by CustomerID
```

행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1
8	12354	1
9	12355	1
10	12356	3
11	12357	1
12	12358	2
13	12359	6
14	12360	3
15	12361	1
16	12362	13

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
select CustomerID, sum(Quantity) as  item_cnt
from sincere-burner-456110-d0.modulabs_project.data
```

```
group by CustomerID
```

[결과 이미지를 넣어주세요]

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

```
-- 전체 거래 건수 계산과 구매한 아이템의 총 수량 계산의 결과를 합쳐서 user_rf라는 테이블에 저장
create or replace table sincere-burner-456110-d0.modulabs_project.user_rf as
-- (1) 전체 거래 건수 계산
with purchase_cnt as (
    select CustomerID, count(distinct InvoiceNo) as purchase_cnt
    from sincere-burner-456110-d0.modulabs_project.data
    group by CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt as (
     select CustomerID, sum(Quantity) as  item_cnt
     from sincere-burner-456110-d0.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 sincere-burner-456110-d0.modulabs_project.user_r as ur
on pc.CustomerID = ur.CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13436	1	76	1
3	15520	1	314	1
4	14569	1	79	1
5	13298	1	96	1
6	14204	1	72	2
7	15195	1	1404	2
8	15471	1	256	2
9	16569	1	93	3
10	17914	1	457	3
11	12442	1	181	3
12	12650	1	250	3
13	15318	1	642	3
14	16528	1	171	3
15	15992	1	17	3
16	14578	1	240	3
17	12478	1	233	3
18	14219	1	78	4
19	15097	1	170	4
20	16597	1	184	4

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
select CustomerID, round(sum(UnitPrice),1)
from sincere-burner-456110-d0.modulabs_project.data
group by CustomerID
```

행	CustomerID	f0_
1	12346	2.1
2	12347	481.2
3	12348	18.7
4	12349	305.1
5	12350	25.3
6	12352	330.5
7	12353	24.3
8	12354	261.2
9	12355	54.7
10	12356	170.9
11	12357	438.7
12	12358	77.2
13	12359	2210.1
14	12360	337.9
15	12361	18.3
16	12362	930.3

- **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 sincere-burner-456110-d0.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) as user_average
    from sincere-burner-456110-d0.modulabs_project.user_rf as rf
    left join (
        select CustomerID, round(sum(UnitPrice * Quantity),1) as user_total
        from sincere-burner-456110-d0.modulabs_project.data
        group by CustomerID
    ) as ut
    on rf.CustomerID = ut.CustomerID
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	795.0
2	15520	1	314	1	343.5	344.0
3	13298	1	96	1	360.0	360.0
4	14569	1	79	1	227.4	227.0
5	13436	1	76	1	196.9	197.0
6	15195	1	1404	2	3861.0	3861.0
7	15471	1	256	2	454.5	455.0
8	14204	1	72	2	150.6	151.0
9	17914	1	457	3	329.4	329.0
10	14578	1	240	3	168.6	169.0
11	16569	1	93	3	124.2	124.0
12	15992	1	17	3	42.0	42.0
13	15318	1	642	3	312.6	313.0
14	12442	1	181	3	144.1	144.0
15	16528	1	171	3	244.4	244.0
16	12650	1	250	3	242.4	242.0

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select *
from sincere-burner-456110-d0.modulabs_project.user_rfm
```


행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	795.0
2	15520	1	314	1	343.5	344.0
3	13298	1	96	1	360.0	360.0
4	14569	1	79	1	227.4	227.0
5	13436	1	76	1	196.9	197.0
6	15195	1	1404	2	3861.0	3861.0
7	15471	1	256	2	454.5	455.0
8	14204	1	72	2	150.6	151.0
9	17914	1	457	3	329.4	329.0
10	14578	1	240	3	168.6	169.0
11	16569	1	93	3	124.2	124.0
12	15992	1	17	3	42.0	42.0
13	15318	1	642	3	312.6	313.0
14	12442	1	181	3	144.1	144.0
15	16528	1	171	3	244.4	244.0
16	12650	1	250	3	242.4	242.0

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**

```
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;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15657	1	24	22	30.0	30.0	1
2	17331	1	16	123	175.2	175.0	1
3	18174	1	50	7	104.0	104.0	1
4	17948	1	144	147	358.6	359.0	1
5	13302	1	5	155	63.8	64.0	1
6	16765	1	4	294	34.0	34.0	1
7	17923	1	50	282	207.5	208.0	1
8	16323	1	50	196	207.5	208.0	1
9	15316	1	100	326	165.0	165.0	1
10	17752	1	192	359	80.6	81.0	1
11	14424	1	48	17	322.1	322.0	1
12	16061	1	-1	269	-29.9	-30.0	1
13	18233	1	4	325	440.0	440.0	1
14	15510	1	2	330	250.0	250.0	1
15	12943	1	-1	301	-3.8	-4.0	1
16	15753	1	144	304	79.2	79.0	1
17	15940	1	4	311	35.8	36.0	1
18	17307	1	-144	365	-152.6	-153.0	1
19	13747	1	8	373	79.6	80.0	1
20	15070	1	36	372	106.2	106.0	1

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_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
      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 `sincere-burner-456110-d0.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS cancel_frequency
  FROM `sincere-burner-456110-d0.modulabs_project.data`
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(cancel_frequency/total_transactions, 2) AS cancel_rate
FROM `sincere-burner-456110-d0.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

행		recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	2	261	25.5	26.0	1	0.0	1	0	0.0
2	144	304	79.2	79.0	1	0.0	1	0	0.0
3	101	38	43.5	44.0	2	0.0	1	0	0.0
4	88	267	251.5	252.0	2	0.0	1	0	0.0
5	22	14	100.9	101.0	3	0.0	1	0	0.0
6	31	267	62.9	63.0	4	0.0	1	0	0.0
7	158	120	252.1	252.0	5	0.0	1	0	0.0
8	74	89	103.3	103.0	6	0.0	1	0	0.0
9	160	122	356.0	356.0	6	0.0	1	0	0.0
10	35	187	110.6	111.0	7	0.0	1	0	0.0
11	120	366	155.6	156.0	10	0.0	1	0	0.0
12	251	108	241.1	241.0	12	0.0	1	0	0.0
13	196	276	197.0	197.0	14	0.0	1	0	0.0
14	74	373	205.9	206.0	14	0.0	1	0	0.0
15	63	33	125.6	126.0	20	0.0	1	0	0.0

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

```
select *
from sincere-burner-456110-d0.modulabs_project.user_data
```

행		recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	2	261	25.5	26.0	1	0.0	1	0	0.0
2	144	304	79.2	79.0	1	0.0	1	0	0.0
3	101	38	43.5	44.0	2	0.0	1	0	0.0
4	88	267	251.5	252.0	2	0.0	1	0	0.0
5	22	14	100.9	101.0	3	0.0	1	0	0.0
6	31	267	62.9	63.0	4	0.0	1	0	0.0
7	158	120	252.1	252.0	5	0.0	1	0	0.0
8	74	89	103.3	103.0	6	0.0	1	0	0.0
9	160	122	356.0	356.0	6	0.0	1	0	0.0
10	35	187	110.6	111.0	7	0.0	1	0	0.0
11	120	366	155.6	156.0	10	0.0	1	0	0.0
12	251	108	241.1	241.0	12	0.0	1	0	0.0
13	196	276	197.0	197.0	14	0.0	1	0	0.0
14	74	373	205.9	206.0	14	0.0	1	0	0.0
15	63	33	125.6	126.0	20	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep : 학습문제를 풀려고 노력한점

Problem : 학습문제가 어려웠던 점

Try : 학습문제를 풀기위해 sql 숙련도를 향상해야 할 필요가 있음