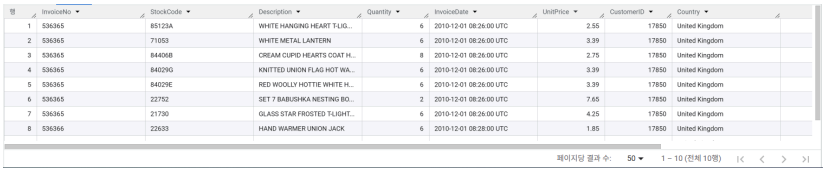
고객을 세그먼테이션하자 [프로젝트] - 유정하

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
select *
from `modulabs_project.data`
limit 10;
```

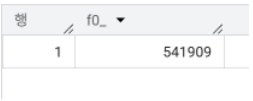


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom

페이지당 결과 수: 50 1 – 10 (전체 10행)

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
select count(*)
from `modulabs_project.data`;
```

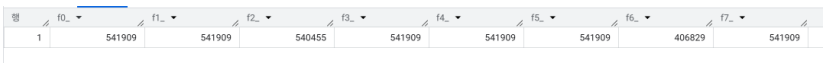


행	f0_
1	541909

데이터 수 세기

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

```
select
  count(InvoiceNo),
  count(StockCode),
  count(Description),
  count(Quantity),
  count(InvoiceDate),
  count(UnitPrice),
  count(CustomerID),
  count(Country)
from `modulabs_project.data`;
```



행	f0_	f1_	f2_	f3_	f4_	f5_	f6_	f7_
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**

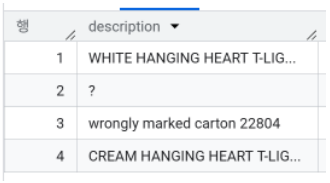
```
SELECT
   'InvoiceNo' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `modulabs_project.data`

UNION ALL
SELECT
   'StockCode',
   ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `modulabs_project.data`

UNION ALL
SELECT
   'Description',
   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `modulabs_project.data`

UNION ALL
SELECT
   'Quantity',
   ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `modulabs_project.data`

UNION ALL
SELECT
   'InvoiceDate',
   ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `modulabs_project.data`

UNION ALL
SELECT
   'UnitPrice',
   ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `modulabs_project.data`

UNION ALL
SELECT
   'CustomerID',
   ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `modulabs_project.data`

UNION ALL
SELECT
   'Country',
   ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `modulabs_project.data`;
```

행	column_name	missing_percenta...
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 project_name.modulabs_project.data
where stockcode = '85123A';
```

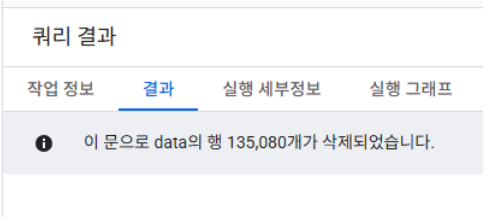


행	description
1	WHITE HANGING HEART T-LIG...
2	?
3	wrongly marked carton 22804
4	CREAM HANGING HEART T-LIG...

결측치 처리

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

```
delete from `modulabs_project.data`
where customerid is null or description is null;
```



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
  SUM(cnt - 1) AS duplicated_rows
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS cnt
  FROM `modulabs_project.data`
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
```

```
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  HAVING cnt > 1
);
```

행	duplicated_rows
1	5225

중복값 처리

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

```
create or replace table `modulabs_project.data` as
select distinct *
from `modulabs_project.data`;
```

행	before_cnt	after_cnt
1	406829	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```
select
  count(distinct invoiceno)
from `modulabs_project.data`;
```

행	f0_
1	22190

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

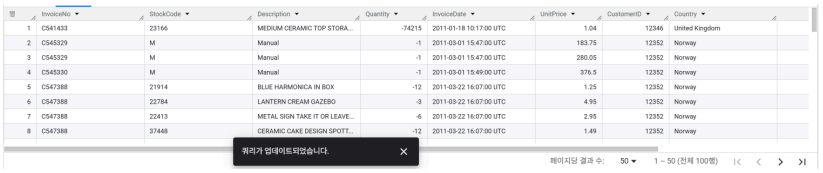
```
select distinct invoiceno
from `modulabs_project.data`
limit 100;
```

행	invoiceno
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180

(밑에 더 있음)

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

```
select *
from `modulabs_project.data`
where invoiceno like 'C%'
limit 100;
```



- **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`;
```

행	f0_
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
select count(distinct stockcode)
from `modulabs_project.data`;
```

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;
```

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 `modulabs_project.data`
)
where number_count <= 1
order by number_count;
```

행	stockcode	number_count
1	POST	0
2	M	0
3	D	0
4	BANK CHARGES	0
5	PADS	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)

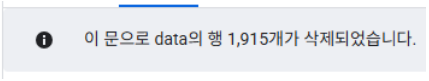
```
with StockCodeCounts as (
  select
    stockcode,
    length(stockcode)
      - length(regexp_replace(stockcode, r'[0-9]', ''))
  from `modulabs_project.data`
)
select
  round(
    (select count(*) from stockcodecounts where number_count <= 1)
    /
    (select count(*) from stockcodecounts)
```

```
    * 100,
    2
  );
```

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,
      length(stockcode)
        - length(regexp_replace(stockcode, r'[0-9]', '')) as number_count
    from `modulabs_project.data`
  )
  where number_count <= 1
);
```



Description 살펴보기

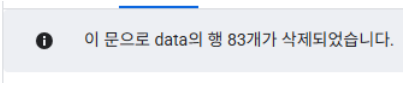
- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
select
  description,
  count(*) as description_cnt
from
  `modulabs_project.data`
group by
  description
order by
  description_cnt desc
limit 30;
```

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 D...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE ...	1062

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
delete
from `modulabs_project.data`
where description in (
  'High Resolution Image',
  'Next Day Carriage'
);
```



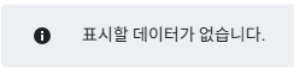
description
BAG 250g SWIRLY MARBLES
3 TRADITIONAI BISCUIT CUTTERS SET
BAG 125g SWIRLY MARBLES
POLYESTER FILLER PAD 30CMx30CM
BAG 500g SWIRLY MARBLES
POLYESTER FILLER PAD 45×45cm
POLYESTER FILLER PAD 40×40cm
ESSENTIAL BALM 3.5g TIN IN ENVELOPE
FRENCH BLUE METAL DOOR SIGN No
POLYESTER FILLER PAD 45×30cm
NUMBER TILE VINTAGE FONT No
NUMBER TILE COTTAGE GARDEN No
POLYESTER FILLER PAD 65CMx65CM
FLOWERS HANDBAG blue and orange
"FOLK ART GREETING CARD,pack/12"
THE KING GIFT BAG 25×24×12cm
POLYESTER FILLER PAD 60×40cm

- **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`;
```

```
select distinct description
from `modulabs_project.data`
where regexp_contains(description, r'[a-z]');
```

를 실행했을 때



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`;
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757405...

- **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_quantity
from `modulabs_project.data`
where unitprice = 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 `modulabs_project.data` as
select *
from `modulabs_project.data`
where unitprice > 0;
```

```
select
  count(*) as cnt_quantity
from `modulabs_project.data`
where unitprice = 0;
```

를 실행했을 때

행	cnt_quantity
1	0

11-7. RFM 스코어

Recency

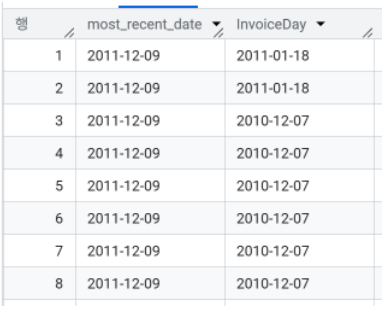
- `InvoiceDate` **컬럼을 연월일 자료형으로 변경하기**

```
select
  date(invoicedate) as invoiceday,
  *
from `modulabs_project.data`;
```

행	invoiceday	InvoiceNo
1	2011-01-18	541431
2	2011-01-18	C541433
3	2010-12-07	537626
4	2010-12-07	537626
5	2010-12-07	537626
6	2010-12-07	537626
7	2010-12-07	537626
8	2010-12-07	537626

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

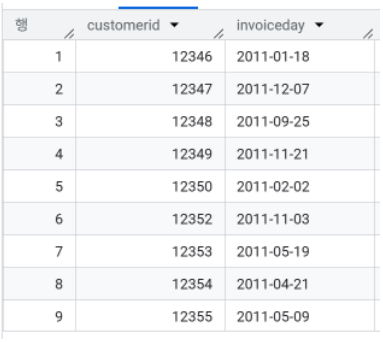
```
SELECT
  (SELECT MAX(DATE(InvoiceDate)) FROM `modulabs_project.data`) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `modulabs_project.data`;
```



행	most_recent_date	InvoiceDay
1	2011-12-09	2011-01-18
2	2011-12-09	2011-01-18
3	2011-12-09	2010-12-07
4	2011-12-09	2010-12-07
5	2011-12-09	2010-12-07
6	2011-12-09	2010-12-07
7	2011-12-09	2010-12-07
8	2011-12-09	2010-12-07

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

```
select
   customerid,
   max(date(invoicedate)) as invoiceday
from `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

- **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	12473	53
2	12508	26
3	12530	59
4	12659	29
5	12676	24
6	12752	81
7	12943	301

- **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
);
```

행	CustomerID	recency
1	16626	0
2	12518	0
3	15311	0
4	12680	0
5	14446	0
6	16705	0
7	17428	0
8	16446	0
9	17581	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
select
   customerid,
   count(distinct invoiceno) as purchase_cnt
from `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

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

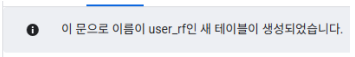
```
select
  customerid,
  sum(quantity) as item_cnt
from `modulabs_project.data`
group by customerid;
```

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
9	12355	240

- **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
with purchase_cnt as (
  select
    customerid,
    count(distinct invoiceno) as purchase_cnt
  from `modulabs_project.data`
  group by customerid
),
item_cnt as (
  select
    customerid,
    sum(quantity) as item_cnt
  from `modulabs_project.data`
  group by customerid
)

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;
```



행	customerid	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	15520	1	314	1
3	14569	1	79	1
4	13436	1	76	1
5	13298	1	96	1
6	15471	1	256	2
7	14204	1	72	2
8	15195	1	1404	2
9	17914	1	457	3

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
select
  customerid,
  round(sum(quantity * unitprice), 1) as user_total
from `modulabs_project.data`
group by customerid;
```

행	customerid	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4

- **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,
  round(ut.user_total / rf.purchase_cnt, 0) as user_average
from `modulabs_project.user_rf` rf
left join (
  select
```

```
    customerid,
    round(sum(quantity * unitprice), 0) as user_total
  from `modulabs_project.data`
  group by customerid
) ut
on rf.customerid = ut.customerid;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	14569	1	79	1	227.0	227.0
3	15520	1	314	1	343.0	343.0
4	13436	1	76	1	197.0	197.0
5	13298	1	96	1	360.0	360.0
6	14204	1	72	2	151.0	151.0
7	15195	1	1404	2	3861.0	3861.0
8	15471	1	256	2	454.0	454.0
9	12478	1	233	3	546.0	546.0

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select *
from `modulabs_project.user_rfm`;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	14569	1	79	1	227.0	227.0
3	15520	1	314	1	343.0	343.0
4	13436	1	76	1	197.0	197.0
5	13298	1	96	1	360.0	360.0
6	14204	1	72	2	151.0	151.0
7	15195	1	1404	2	3861.0	3861.0
8	15471	1	256	2	454.0	454.0
9	12478	1	233	3	546.0	546.0

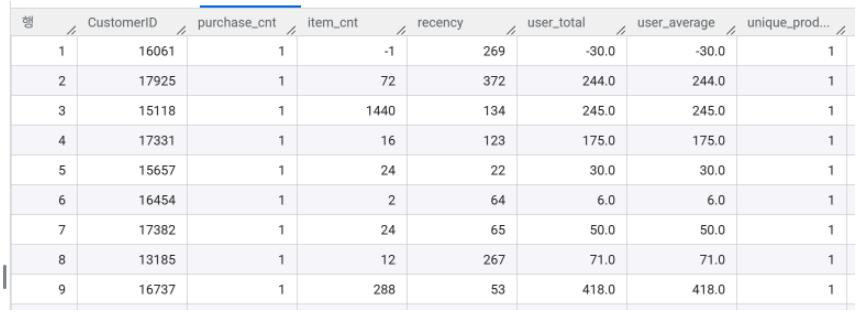
11-8. 추가 Feature 추출

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 modu-480005.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;
```

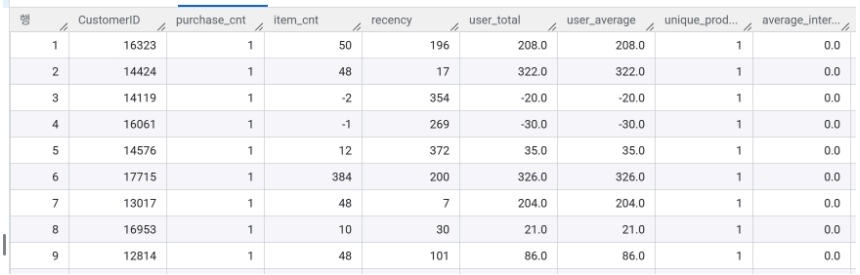


행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...
1	16061	1	-1	269	-30.0	-30.0	1
2	17925	1	72	372	244.0	244.0	1
3	15118	1	1440	134	245.0	245.0	1
4	17331	1	16	123	175.0	175.0	1
5	15657	1	24	22	30.0	30.0	1
6	16454	1	2	64	6.0	6.0	1
7	17382	1	24	65	50.0	50.0	1
8	13185	1	12	267	71.0	71.0	1
9	16737	1	288	53	418.0	418.0	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 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;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...
1	16323	1	50	196	208.0	208.0	1	0.0
2	14424	1	48	17	322.0	322.0	1	0.0
3	14119	1	-2	354	-20.0	-20.0	1	0.0
4	16061	1	-1	269	-30.0	-30.0	1	0.0
5	14576	1	12	372	35.0	35.0	1	0.0
6	17715	1	384	200	326.0	326.0	1	0.0
7	13017	1	48	7	204.0	204.0	1	0.0
8	16953	1	10	30	21.0	21.0	1	0.0
9	12814	1	48	101	86.0	86.0	1	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`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

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

total_transac...	cancel_frequ...	cancel_rate
2	1	0.5
2	1	0.5
182	0	0.0
182	0	0.0
182	0	0.0
182	0	0.0

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

```
SELECT *
FROM `modulabs_project.user_data`;
```

	A	B	C	D	E	F	G	H	I	J	K
1	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description	total_transactions	cancel_frequency	cancel_rate
2	541431	23166	74215	2011-01-18 10:01:00.000000 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE JAR	2	1	0.5
3	C541433	23166	-74215	2011-01-18 10:17:00.000000 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE JAR	2	1	0.5
4	537626	22771	12	2010-12-07 14:57:00.000000 UTC	1.25	12347	Iceland	CLEAR DRAWER KNOB ACRYLIC EDWARDIAN	182	0	0
5	537626	85232D	3	2010-12-07 14:57:00.000000 UTC	4.95	12347	Iceland	SET/3 DECOUPAGE STACKING TINS	182	0	0
6	537626	22774	12	2010-12-07 14:57:00.000000 UTC	1.25	12347	Iceland	RED DRAWER KNOB ACRYLIC EDWARDIAN	182	0	0
7	537626	22727	4	2010-12-07 14:57:00.000000 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE RED	182	0	0
8	537626	85116	12	2010-12-07 14:57:00.000000 UTC	2.1	12347	Iceland	BLACK CANDELABRA T-LIGHT HOLDER	182	0	0
9	537626	22212	6	2010-12-07 14:57:00.000000 UTC	2.1	12347	Iceland	FOUR HOOK WHITE LOVEBIRDS	182	0	0

회고

[회고 내용을 작성해주세요]

Keep : 빅쿼리를 통해 테이블들을 생성하고 변환하여 sql 데이터 전체 흐름을 이해한 점

Problem : 구체적으로 어떻게 작성해야하는지 잘 모르거나 헷갈리던 점들이 존재했음

Try : 콤마같은 것 빼먹지 않는 습관 들이기, sql 더 잘 알고 더 잘 할 수 있도록 노력하기