250708 고객을 세그먼테이션하자! [프로젝트]

RFM 분석

- RFM 분석은 구매 최신성(Recency), 구매 빈도(Frequency)와 구매 가치(Monetary)에 따라 고객들을 여러 그룹으로 나누는 세그먼테이션(segmentation) 방법
- **R**ecency
  - 고객이 **마지막으로 구매한 시점**
  - 최근에 구매한 고객들은 더 자주 구매할 가능성이 높기 때문에, 최신성 점수가 높은지를 고려함
- **F**requency
  - 특정 기간 동안 고객이 **얼마나 자주 우리의 제품이나 서비스를 구매하는지**를 반영
  - 빈번하게 구매를 하는 고객은 더 충성도가 높은 고객일 확률이 높기 때문에, 빈도 점수가 높은지를 고려함
- **M**onetary
  - 고객이 **지출한 총 금액**
  - 높은 금액을 지불한 고객일수록 더 가치가 높은 충성 고객일 수 있음
  - 앞으로도 우리 제품과 사이트에 많은 돈을 지불할 수 있는 고객이므로, 가치 점수가 높은지를 함께 고려함

분석 과정

- '고객을 세그먼테이션하자! [파트1 - SQL 실습]'
  - 데이터 불러오기
  - 데이터 전처리
    - 결측치 제거
    - 중복값 처리
    - 오류값 처리
  - RFM 분석
  - 추가 feature 추출

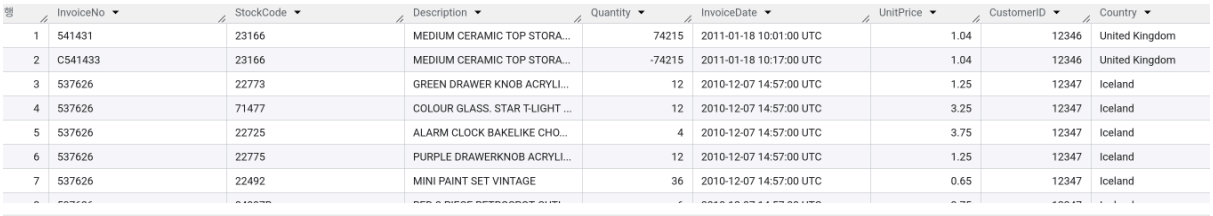
- '고객을 세그먼테이션하자! [파트2 - 파이썬 실습]'
  - 이상 데이터 처리
  - 변수간 상관관계 분석
  - 피처 스케일링
  - 차원 축소
  - K-Means 클러스터링
  - 시각화 및 결과 분석
    - 고객 세그먼테이션을 통한 인사이트와 전략

데이터 구성

컬럼명	설명
InvoiceNo	각각의 고유한 거래를 나타내는 코드. 이 코드가 'C'라는 글자로 시작한다면, 취소를 나타냄 하나의 거래에 여러 개의 제품이 함께 구매되었다면, 1개의 InvoiceNo에는 여러 개의 StockCode가 연결되어 있음
StockCode	각각의 제품에 할당된 고유 코드
Description	각 제품에 대한 설명
Quantity	거래에서 제품을 몇 개 구매했는지에 대한 단위 수
InvoiceDate	거래가 일어난 날짜와 시간
UnitPrice	제품 당 단위 가격(영국 파운드)
CustomerID	각 고객에게 할당된 고유 식별자 코드
Country	주문이 발생한 국가

데이터 전처리

```
-- 10개 행 데이터만 가져와보기
SELECT *
FROM pelagic-campus-464902-b9.modulabs_project.data
LIMIT 10;
```



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	541431	23166	MEDIUM CERAMIC TOP STORA...	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom
2	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
3	537626	22773	GREEN DRAWER KNOB ACRYLI...	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
4	537626	71477	COLOUR GLASS. STAR T-LIGHT ...	12	2010-12-07 14:57:00 UTC	3.25	12347	Iceland
5	537626	22725	ALARM CLOCK BAKELIKE CHO...	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
6	537626	22775	PURPLE DRAWERKNOB ACRYLI...	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
7	537626	22492	MINI PAINT SET VINTAGE	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland

```
-- 컬럼 별 누락된 값의 비율 계산
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 pelagic-campus-464902-b9.modulabs_project.data UNION ALL
    SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total
    FROM pelagic-campus-464902-b9.modulabs_project.data UNION ALL
    SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total
    FROM pelagic-campus-464902-b9.modulabs_project.data UNION ALL
    SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total
    FROM pelagic-campus-464902-b9.modulabs_project.data UNION ALL
    SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total
    FROM pelagic-campus-464902-b9.modulabs_project.data UNION ALL
    SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total
    FROM pelagic-campus-464902-b9.modulabs_project.data UNION ALL
    SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total
    FROM pelagic-campus-464902-b9.modulabs_project.data UNION ALL
    SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total
    FROM pelagic-campus-464902-b9.modulabs_project.data
) AS column_data;
```

→ 결측치 비율을 보면 CustomerID가 24.93%로 상당이 높음

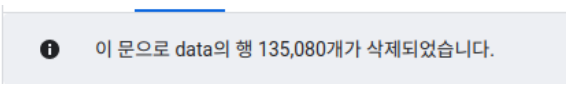
--> 큰 비율의 누락된 값을 다른 값으로 대체하는 것은 분석에 상당한 편향을 주고 노이즈가 될 수 있으므로 CustomerID가 누락된 행 제거

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

- Description 결측치 처리하기

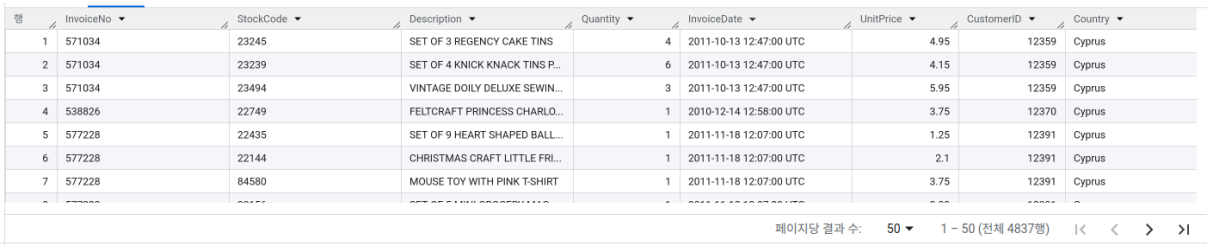
```
-- 아래의 코드를 실행한 결과 같은 StockCode라도 Description이 다른 경우가 있음을
-- 따라서 Description의 결측치를 같은 StockCode의 값으로 대체했을 때 신뢰하기 어
-- Description의 결측 비율이 0.27%로 낮기 때문에 결측행을 제거하기로 함
SELECT DISTINCT Description
FROM pelagic-campus-464902-b9.modulabs_project.data
WHERE StockCode = '85123A';

-- 결측치 처리하기
DELETE FROM pelagic-campus-464902-b9.modulabs_project.data
WHERE CustomerID IS NULL OR Description IS NULL;
```

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



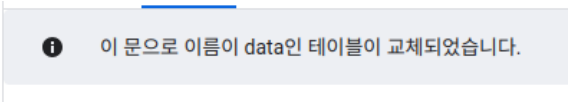
- 중복값 처리하기

```
-- 중복값 확인하기
SELECT *
FROM pelagic-campus-464902-b9.modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPric
HAVING COUNT(*) > 1;
→ 동일한 거래 시간을 포함한 동일한 행은 데이터 오류일 가능성이 높으므로 분석 결과어

-- 중복값 처리하기
CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_project.d
SELECT DISTINCT *
FROM pelagic-campus-464902-b9.modulabs_project.data;

-- 중복 제거 후 데이터 개수
SELECT COUNT(*)
FROM pelagic-campus-464902-b9.modulabs_project.data;
```



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	571034	23245	SET OF 3 REGENCY CAKE TINS	4	2011-10-13 12:47:00 UTC	4.95	12359	Cyprus
2	571034	23239	SET OF 4 KNICK KNACK TINS P...	6	2011-10-13 12:47:00 UTC	4.15	12359	Cyprus
3	571034	23494	VINTAGE DOILY DELUXE SEWIN...	3	2011-10-13 12:47:00 UTC	5.95	12359	Cyprus
4	538826	22749	FELTCRAFT PRINCESS CHARLO...	1	2010-12-14 12:58:00 UTC	3.75	12370	Cyprus
5	577228	22435	SET OF 9 HEART SHAPED BALL...	1	2011-11-18 12:07:00 UTC	1.25	12391	Cyprus
6	577228	22144	CHRISTMAS CRAFT LITTLE FRI...	1	2011-11-18 12:07:00 UTC	2.1	12391	Cyprus
7	577228	84580	MOUSE TOY WITH PINK T-SHIRT	1	2011-11-18 12:07:00 UTC	3.75	12391	Cyprus

페이지당 결과 수: 50 1 – 50 (전체 4837행)



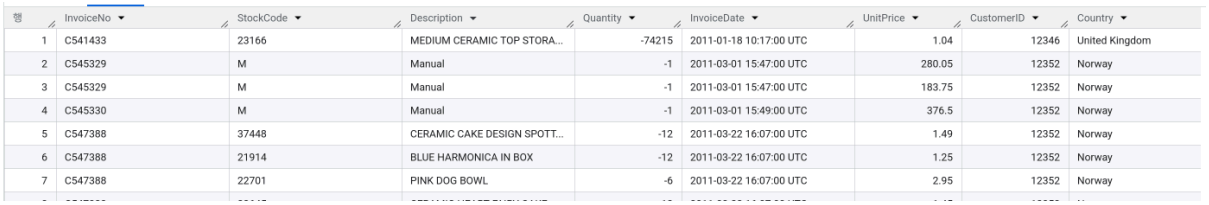
행	f0_
1	401604

- InvoiceNo 컬럼 살펴보기

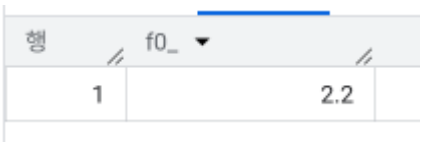
```
-- 고유(unique)한 InvoiceNo 출력하기
SELECT DISTINCT InvoiceNo
FROM pelagic-campus-464902-b9.modulabs_project.data
LIMIT 100;

-- InvoiceNo에 C로 시작하는 값들이 있음 -> 취소된 거래를 의미하므로 해당 행들 살펴
SELECT *
```

```
FROM pelagic-campus-464902-b9.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```
→ InvoiceNo에 C로 시작하는 값들의 특징을 보면 Quantity가 음수임을 알 수 있음

```
-- 구매 건 상태가 Canceled 인 데이터의 비율(%) 구해보기
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)
FROM pelagic-campus-464902-b9.modulabs_project.data;
```
→ 2.2%



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
6	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
7	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway



행	f0_
1	2.2

- StockCode 컬럼 살펴보기

```
-- 고유한 StockCode의 개수를 출력
SELECT COUNT(DISTINCT StockCode)
FROM pelagic-campus-464902-b9.modulabs_project.data;
```
→ 3684개

```
-- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력해보기 (상위 10개의 제품 출력)
SELECT StockCode, COUNT(*) AS sell_cnt
FROM pelagic-campus-464902-b9.modulabs_project.data
GROUP BY StockCodeORDER BY sell_cnt DESCLIMIT 10;
```
→ 제품 코드는 대부분 5~6자리의 숫자와 문자 조합으로 구성되어 있는 반면, 'POST'와 같은 몇 가지 이상한 코드도 있음

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

```
-- 'POST'와 같은 비정상적인 항목들은 숫자가 0개 포함되어 있으므로 StockCode 내 숫자의 개수를 살펴보기
WITH UniqueStockCodes AS (
    SELECT DISTINCT StockCode
    FROM pelagic-campus-464902-b9.modulabs_project.data)
SELECT  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,  COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC;
→ 쿼리 실행 결과 8개를 제외하곤 StockCode에 5개의 숫자들이 포함되어 있음을 알 수 있음
/* - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) : StockCode에서 숫자를 모두 ''로 대체한 후 길이를 재는 코드
-> 문자만 남기고 그 길이를 재는 것
- 빅쿼리에서는 쿼리 최적화 과정을 통해 GROUP BY에 SELECT 절에 정의한 별칭을 사용해도 오류가 나지 않음
이는 DBMS마다 다르기 때문에 정확하게 사용하려면 별칭 말고 LENGTH(StockCode)
- LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) 이 부분을 group by에 써줘야 함
-> 쿼리 실행 순서가 SELECT 절이 GROUP BY 절보다 뒤에 실행되기 때문*/
```

행	number_count	stock_cnt
1	5	3676
2	0	7
3	1	1

```
-- 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지를 확인하기
SELECT DISTINCT StockCode, number_count
FROM (
   SELECT StockCode,
   LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]',
'')) AS number_count
   FROM pelagic-campus-464902-b9.modulabs_project.data)
WHERE number_count <=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

```
-- 제품과 관련되지 않은 거래 기록을 제거하기
DELETE FROM pelagic-campus-464902-b9.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 pelagic-campus-464902-b9.modulabs_project.data  )
WHERE number_count <=1);
```

- Description 살펴보기

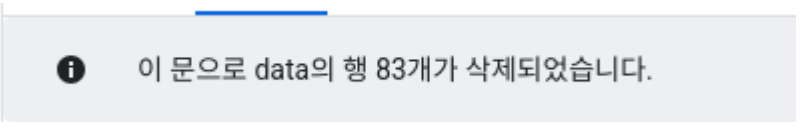
```
-- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기
SELECT Description, COUNT(Description) AS description_cnt
FROM pelagic-campus-464902-b9.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

행	Description	description_cnt
1	WHITE HANGING HEART T-LIGHT HOLDER	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORNAMENT	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY DESIGN	1224

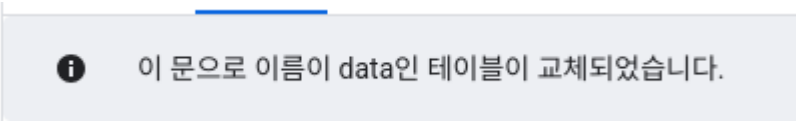
```
-- 대소문자가 혼합된 값이 있는지 확인하기
SELECT DISTINCT Description
FROM pelagic-campus-464902-b9.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');  -- 소문자 포함 여부 확인
```

행	Description
1	BAG 250g SWIRLY MARBLES
2	3 TRADITIONAl BISCUIT CUTTERS SET
3	BAG 125g SWIRLY MARBLES
4	POLYESTER FILLER PAD 30CMx30CM
5	BAG 500g SWIRLY MARBLES
6	POLYESTER FILLER PAD 45x45cm
7	POLYESTER FILLER PAD 40x40cm

```
-- 서비스 관련 정보를 포함하는 행들을 제거
DELETE FROM pelagic-campus-464902-b9.modulabs_project.data
WHERE Description IN ('High Resolution Image', 'Next Day Carriage');
```



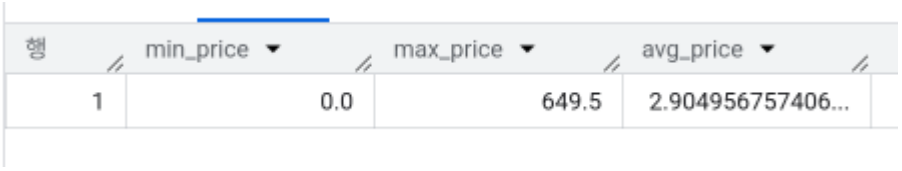

```
-- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화하기
CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_proje
ct.data AS
SELECT
    -- 테이블의 모든 컬럼들을 가져오되, Description은 제외하라는 의미
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM pelagic-campus-464902-b9.modulabs_project.data;
```



- UnitPrice 살펴보기

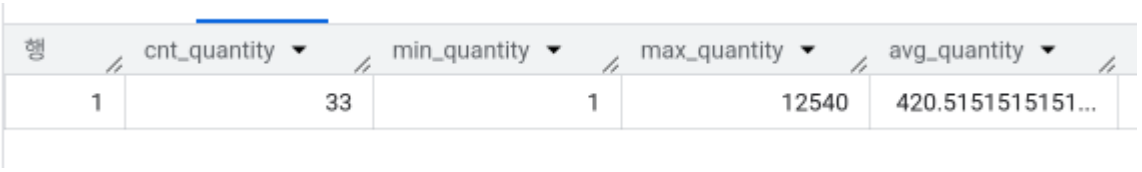
```
-- UnitPrice의 최솟값, 최댓값, 평균구하기
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG
```

```
(UnitPrice) AS avg_price
FROM pelagic-campus-464902-b9.modulabs_project.data;
```



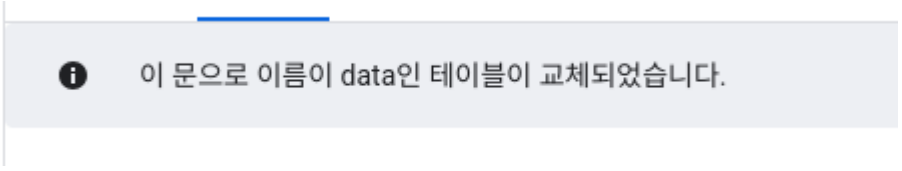
행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

```
-- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균구하기
SELECT COUNT(*) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Q
uantity) AS max_quantity, AVG(Quantity) AS avg_quantity
FROM pelagic-campus-464902-b9.modulabs_project.data
WHERE UnitPrice = 0;
```



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

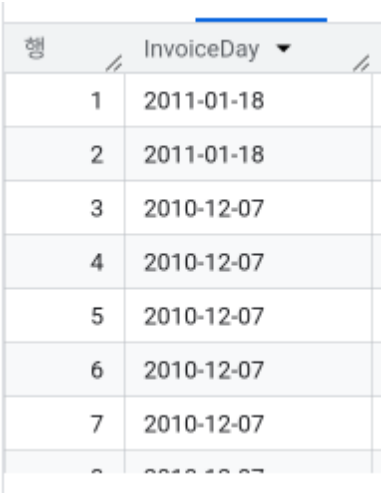
```
-- 무료 제품이라기 보다는 데이터 오류로 보이므로 제거하기
CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_proje
ct.data AS
SELECT *
FROM pelagic-campus-464902-b9.modulabs_project.data
WHERE UnitPrice != 0;
```



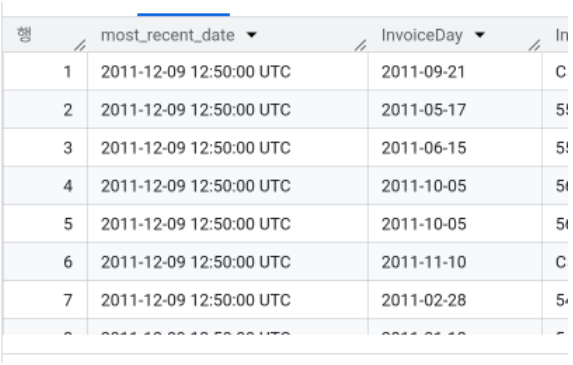
RFM 스코어 구하기

1. **Recency**

```
-- InvoiceDate 컬럼을 연월일 자료형으로 변경하기
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM pelagic-campus-464902-b9.modulabs_project.data

-- 최근 구매 일자를 MAX() 함수로 찾아보기
SELECT MAX(InvoiceDate) OVER() AS most_recent_date,
     DATE(InvoiceDate) AS InvoiceDay, *
FROM pelagic-campus-464902-b9.modulabs_project.data;
```



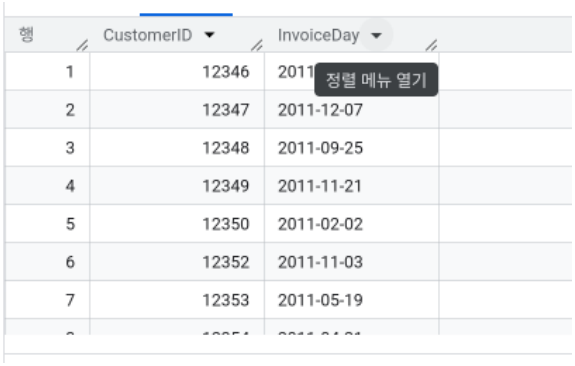
행	InvoiceDay
1	2011-01-18
2	2011-01-18
3	2010-12-07
4	2010-12-07
5	2010-12-07
6	2010-12-07
7	2010-12-07



행	most_recent_date	InvoiceDay	In
1	2011-12-09 12:50:00 UTC	2011-09-21	C
2	2011-12-09 12:50:00 UTC	2011-05-17	5
3	2011-12-09 12:50:00 UTC	2011-06-15	5
4	2011-12-09 12:50:00 UTC	2011-10-05	5
5	2011-12-09 12:50:00 UTC	2011-10-05	5
6	2011-12-09 12:50:00 UTC	2011-11-10	C
7	2011-12-09 12:50:00 UTC	2011-02-28	5

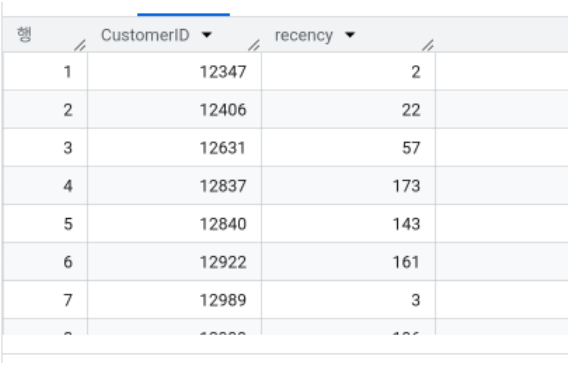
```
-- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기
SELECT
   CustomerID,
   DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM pelagic-campus-464902-b9.modulabs_project.data
GROUP BY CustomerID;

-- 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 pelagic-campus-464902-b9.modulabs_project.data
 GROUP BY CustomerID
);
```



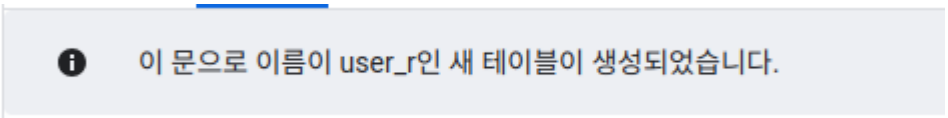
행	CustomerID	InvoiceDay
1	12346	2011
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19

정렬 메뉴 열기



행	CustomerID	recency
1	12347	2
2	12406	22
3	12631	57
4	12837	173
5	12840	143
6	12922	161
7	12989	3

```
-- 지금까지의 결과를 user_r이라는 이름의 테이블로 저장하기
CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_proje
ct.user_r AS
SELECT
 CustomerID,
 EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
 SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
 FROM pelagic-campus-464902-b9.modulabs_project.data
 GROUP BY CustomerID
);
```



2. **Frequency**

```
-- 전체 거래 건수 계산 -> 고객마다 고유한 InvoiceNo의 수
SELECT
 CustomerID,
```

```
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM pelagic-campus-464902-b9.modulabs_project.data
GROUP BY CustomerID;

-- 구매한 아이템의 총 수량 계산 -> 각 고객 별로 구매한 아이템의 총 수량
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM pelagic-campus-464902-b9.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

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

```
-- '1. 전체 거래 건수 계산'과 '2. 구매한 아이템의 총 수량 계산'의 결과를 합쳐서 user_rf라는 이름의 테이블에 저장하기
-- CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM pelagic-campus-464902-b9.modulabs_project.data
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT CustomerID, SUM(Quantity) AS item_cnt
  FROM pelagic-campus-464902-b9.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 pelagic-campus-464902-b9.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency
1	12346	2	0	325
2	12347	7	2458	2
3	12348	4	2332	75
4	12349	1	630	18
5	12350	1	196	310
6	12352	8	463	36
7	12353	1	20	204

3. **Monetary**

```
-- 고객별 총 지출액 계산(소수점 첫째 자리에서 반올림)
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity)) AS user_total
FROM pelagic-campus-464902-b9.modulabs_project.data
GROUP BY CustomerID;
```

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

```
-- 고객별 평균 거래 금액 계산
-- 1) data 테이블을 user_rf 테이블과 조인(LEFT JOIN) 한 후,
-- 2) purchase_cnt로 나누어서 3) user_rfm 테이블로 저장
CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_proje
ct.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 pelagic-campus-464902-b9.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT CustomerID, ROUND(SUM(UnitPrice * Quantity)) AS user_total
  FROM pelagic-campus-464902-b9.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;


-- 최종 user_rfm 테이블 출력
SELECT *
FROM pelagic-campus-464902-b9.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	13298	1	96	1	360.0	360.0
3	15520	1	314	1	343.0	343.0
4	13436	1	76	1	197.0	197.0
5	14569	1	79	1	227.0	227.0
6	15471	1	256	2	454.0	454.0
7	15195	1	1404	2	3861.0	3861.0

추가 Feature 추출

1. **구매하는 제품의 다양성**

```
-- 1) 고객 별로 구매한 상품들의 고유한 수를 계산
-- 2) user_rfm 테이블과 결과를 합치고
-- 3) user_data라는 이름의 테이블에 저장
CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_proje
ct.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    -- 고객 별로 구매한 상품들의 고유한 수
    COUNT(DISTINCT StockCode) AS unique_products
  FROM pelagic-campus-464902-b9.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM pelagic-campus-464902-b9.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	12713	1	505	0	795.0	795.0	37
2	13298	1	96	1	360.0	360.0	2
3	15520	1	314	1	343.0	343.0	18
4	13436	1	76	1	197.0	197.0	12
5	14569	1	79	1	227.0	227.0	10
6	15471	1	256	2	454.0	454.0	67
7	15195	1	1404	2	3861.0	3861.0	1

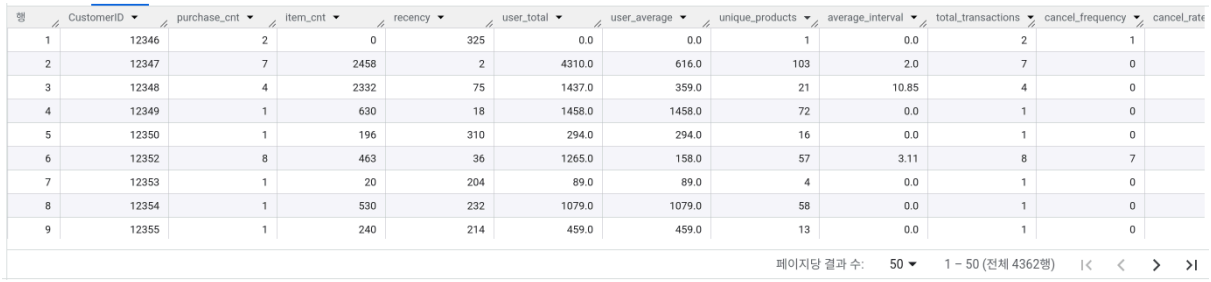
2. 평균 구매 주기

```
--평균 구매 소요 일수(고객들의 구매와 구매 사이의 기간이 평균적으로 몇 일인지를 보여주는 값)를 계산하고, 그 결과를 user_data에 통합
CREATE OR REPLACE TABLE pelagic-campus-464902-b9.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(현재 구매일에서 직전 구매일을 뺀 값)
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
    FROM
      pelagic-campus-464902-b9.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM pelagic-campus-464902-b9.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_products	average_interval	average_interval_1
1	13747	1	8	373	80.0	80.0	1	0.0	0.0
2	14090	1	72	324	76.0	76.0	1	0.0	0.0
3	15562	1	39	351	135.0	135.0	1	0.0	0.0
4	18068	1	6	289	102.0	102.0	1	0.0	0.0
5	13302	1	5	155	64.0	64.0	1	0.0	0.0
6	16323	1	50	196	208.0	208.0	1	0.0	0.0
7	15524	1	4	24	440.0	440.0	1	0.0	0.0

3. 구매 취소 경향성

```
-- 취소 빈도 : 고객 별로 취소한 거래의 총 횟수
-- 취소 비율 : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
-- 취소 빈도와 취소 비율을 계산하고 그 결과를 user_data에 통합하기 (취소 비율은 소수점 두번째 자리까지 구하기)

CREATE OR REPLACE TABLE pelagic-campus-464902-b9.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions, -- 전체 거래 횟수
    COUNTIF(InvoiceNo LIKE 'C%') AS cancel_frequency
  FROM pelagic-campus-464902-b9.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate
FROM pelagic-campus-464902-b9.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
order by CustomerID;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	12346	2	0	325	0.0	0.0	1	0.0	2	1	
2	12347	7	2458	2	4310.0	616.0	103	2.0	7	0	
3	12348	4	2332	75	1437.0	359.0	21	10.85	4	0	
4	12349	1	630	18	1458.0	1458.0	72	0.0	1	0	
5	12350	1	196	310	294.0	294.0	16	0.0	1	0	
6	12352	8	463	36	1265.0	158.0	57	3.11	8	7	
7	12353	1	20	204	89.0	89.0	4	0.0	1	0	
8	12354	1	530	232	1079.0	1079.0	58	0.0	1	0	
9	12355	1	240	214	459.0	459.0	13	0.0	1	0	

페이지당 결과 수: 50 1 – 50 (전체 4362행)