고객을 세그먼테이션하자

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM long-ceiling-470102-p4.moulabs_project.data
LIMIT 10
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custo
1	536365	85123A	WHITE HANGING HEART T-LIGH...	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 H...	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-LIGHT ...	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 DOT	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(*)
FROM long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	541909

데이터 수 세기

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

```
SELECT
  COUNT(InvoiceNo) as 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 long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

행	Count_InvoiceNo	Count_StockCode	Count_Description	Count_Quantity	Count_InvoiceDate	Count_UnitPrice	Count_CustomerID	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**

```
-- 사용자 정의 함수 만들어 쓰는게 나을지도..?
SELECT
   'InvoiceNo' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

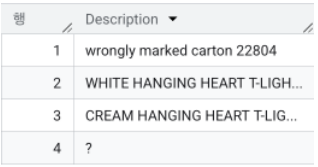
행	column_name	missing_percenta...
1	UnitPrice	0.0
2	InvoiceDate	0.0
3	Quantity	0.0
4	CustomerID	24.93
5	Description	0.27
6	StockCode	0.0
7	Country	0.0
8	InvoiceNo	0.0

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM long-ceiling-470102-p4.moulabs_project.data
WHERE StockCode = '85123A'
```

[결과 이미지를 넣어주세요]



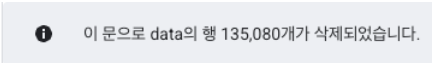
행	Description
1	wrongly marked carton 22804
2	WHITE HANGING HEART T-LIGH...
3	CREAM HANGING HEART T-LIG...
4	?

결측치 처리

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

```
DELETE FROM long-ceiling-470102-p4.moulabs_project.data
WHERE CustomerID is null OR Description ='?'
```

[결과 이미지를 넣어주세요]



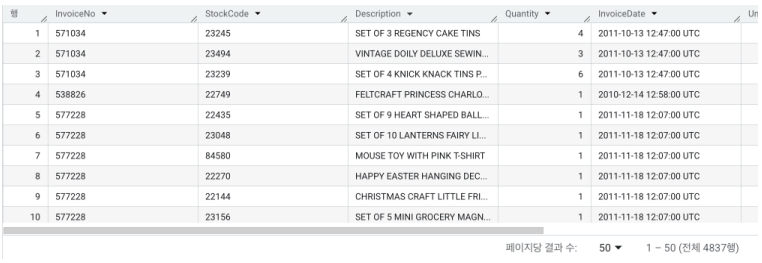
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT *
FROM long-ceiling-470102-p4.moulabs_project.data
GROUP BY InvoiceNo,StockCode,Description,Quantity,UnitPrice,InvoiceDate,CustomerID,Country
HAVING COUNT(*) > 1
```

[결과 이미지를 넣어주세요]



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	571034	23245	SET OF 3 REGENCY CAKE TINS	4	2011-10-13 12:47:00 UTC
2	571034	23494	VINTAGE DOILY DELUXE SEWIN...	3	2011-10-13 12:47:00 UTC
3	571034	23239	SET OF 4 KNICK KNACK TINS P...	6	2011-10-13 12:47:00 UTC
4	538826	22749	FELTCRAFT PRINCESS CHARLO...	1	2010-12-14 12:58:00 UTC
5	577228	22435	SET OF 9 HEART SHAPED BALL...	1	2011-11-18 12:07:00 UTC
6	577228	23048	SET OF 10 LANTERNS FAIRY LI...	1	2011-11-18 12:07:00 UTC
7	577228	84580	MOUSE TOY WITH PINK T-SHIRT	1	2011-11-18 12:07:00 UTC
8	577228	22270	HAPPY EASTER HANGING DEC...	1	2011-11-18 12:07:00 UTC
9	577228	22144	CHRISTMAS CRAFT LITTLE FRI...	1	2011-11-18 12:07:00 UTC
10	577228	23156	SET OF 5 MINI GROCERY MAGN...	1	2011-11-18 12:07:00 UTC

페이지당 결과 수: 50 1 – 50 (전체 4837행)

중복값 처리

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

```
CREATE OR REPLACE TABLE `long-ceiling-470102-p4.moulabs_project.data` AS
SELECT DISTINCT *
FROM long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

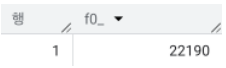
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]



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

```
SELECT InvoiceNo
FROM long-ceiling-470102-p4.moulabs_project.data
LIMIT 100
```

[결과 이미지를 넣어주세요]

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

```
SELECT *
FROM long-ceiling-470102-p4.moulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Ur
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	
5	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	
6	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	
7	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	
8	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	
9	C547388	22645	CERAMIC HEART FAIRY CAKE M...	-12	2011-03-22 16:07:00 UTC	
10	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	

페이지당 결과 수: 50 1 – 50 (전체 100행)

- **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 long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode)
FROM long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	3684

- **어떤 제품이 가장 많이 판매되었는지 보기 위하여 `StockCode` 별 등장 빈도를 출력하기**
  - 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM long-ceiling-470102-p4.moulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

[결과 이미지를 넣어주세요]

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 long-ceiling-470102-p4.moulabs_project.data
)
WHERE number_count in (0,1)
```

[결과 이미지를 넣어주세요]

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)

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM project_name.modulabs_project.data
)
WHERE # [[YOUR QUERY]];
```

0.48%

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `long-ceiling-470102-p4.moulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT
      StockCode,
      (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) AS number_count
    FROM long-ceiling-470102-p4.moulabs_project.data
    WHERE StockCode IS NOT NULL
  )
  WHERE number_count IN (0, 1)
);
```

[결과 이미지를 넣어주세요]

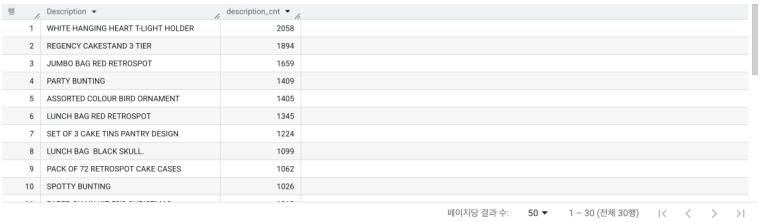
이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM long-ceiling-470102-p4.moulabs_project.data
GROUP BY Description
ORDER BY COUNT(*) DESC
LIMIT 30
```

[결과 이미지를 넣어주세요]



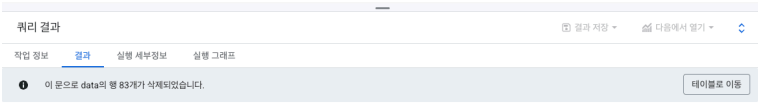
행	Description	description_cnt
1	WHITE HANGING HEART T-LIGHT HOLDER	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORNAMENT	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY DESIGN	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE CASES	1062
10	SPOTTY BUNTING	1026

페이지당 결과 수: 50 1 – 30 (전체 30행)

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM long-ceiling-470102-p4.moulabs_project.data
WHERE Description IN ('Next Day Carriage','High Resolution Image')
```

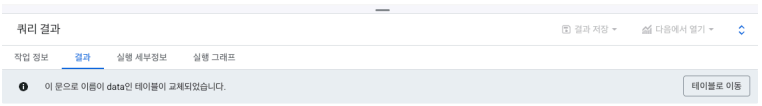
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE long-ceiling-470102-p4.moulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

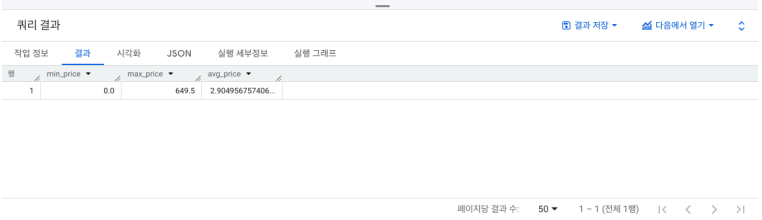


UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM long-ceiling-470102-p4.moulabs_project.data
```

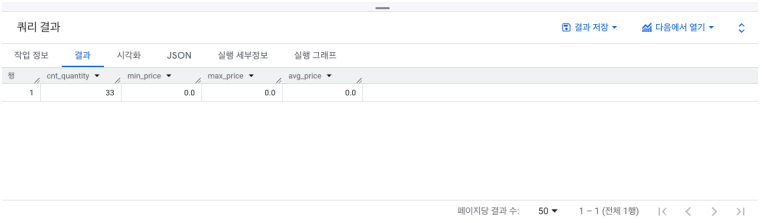
[결과 이미지를 넣어주세요]



- **단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기**

```
SELECT COUNT(*) AS cnt_quantity, MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_pric
e
FROM long-ceiling-470102-p4.moulabs_project.data
WHERE UnitPrice = 0
```

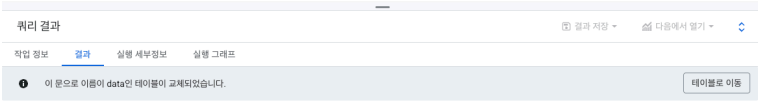
[결과 이미지를 넣어주세요]



- **UnitPrice = 0를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE long-ceiling-470102-p4.moulabs_project.data AS
SELECT *
FROM long-ceiling-470102-p4.moulabs_project.data
WHERE UnitPrice != 0
```

[결과 이미지를 넣어주세요]



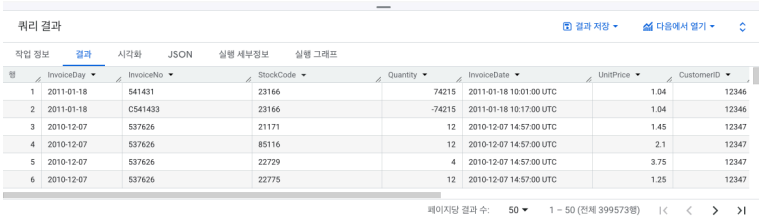
11-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) as InvoiceDay, *
FROM long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(DATE(InvoiceDate)) OVER() as most_recent_date,
  DATE(InvoiceDate) as InvoiceDay,
  *
FROM long-ceiling-470102-p4.moulabs_project.data
```

[결과 이미지를 넣어주세요]

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

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) as InvoiceDay
FROM long-ceiling-470102-p4.moulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

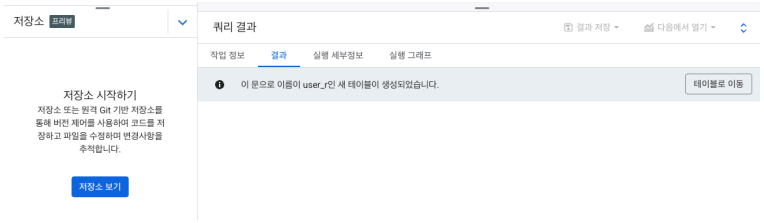
```
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
);

-- 위에 마이너스 연산한 뒤에 EXTRACT를 사용하는 방식보다는 DATE_DFF(END, START, DAY)가 더 안정적일 듯?
-- 물론 두 개의 연산 결과는 모두 같긴 하지만 확장성 및 안정성 고려했을 경우 DATE_DIFF 사용하는 걸로...
WITH per_customer AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM long-ceiling-470102-p4.moulabs_project.data
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
SELECT
  CustomerID,
  DATE_DIFF( MAX(InvoiceDay) OVER (), InvoiceDay, DAY ) AS recency
FROM per_customer
ORDER BY CustomerID;
```

[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE long-ceiling-470102-p4.moulabs_project.user_r AS
WITH per_customer AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM long-ceiling-470102-p4.moulabs_project.data
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
SELECT
  CustomerID,
  DATE_DIFF( MAX(InvoiceDay) OVER (), InvoiceDay, DAY ) AS recency
FROM per_customer
```

[결과 이미지를 넣어주세요]



Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

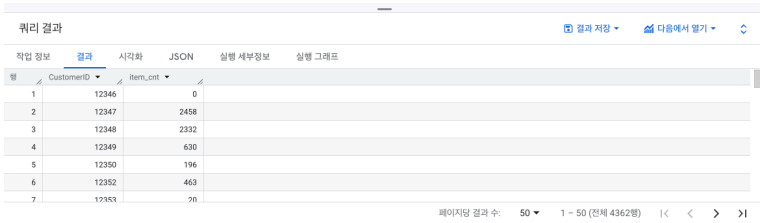
```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo)
FROM long-ceiling-470102-p4.moulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) as item_cnt
FROM long-ceiling-470102-p4.moulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

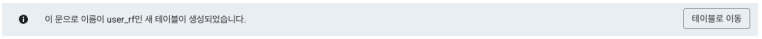


- **전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 `user_rf` 라는 이름의 테이블에 저장하기**

```
CREATE OR REPLACE TABLE long-ceiling-470102-p4.moulabs_project.user_rf AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) as purchase_cnt
  FROM long-ceiling-470102-p4.moulabs_project.data
```

```
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) as item_cnt
  FROM long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13298	1	96	1
3	14569	1	79	1
4	13436	1	76	1
5	15520	1	314	1
6	15195	1	1404	2
7	14204	1	72	2
8	15471	1	256	2
9	12650	1	250	3
10	17914	1	457	3
11	16569	1	93	3

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice)) AS user_total
FROM long-ceiling-470102-p4.moulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

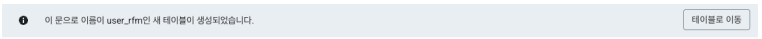
행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.0
4	12349	1458.0
5	12350	294.0
6	12352	1265.0
7	12353	89.0
8	12354	1079.0
9	12355	459.0
10	12356	2487.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

- **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 long-ceiling-470102-p4.moulabs_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 long-ceiling-470102-p4.moulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM long-ceiling-470102-p4.moulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM long-ceiling-470102-p4.moulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

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	14569	1	79	1	227.0	227.0
4	15520	1	314	1	343.0	343.0
5	13436	1	76	1	197.0	197.0
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	16528	1	171	3	244.0	244.0
10	12650	1	250	3	242.0	242.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

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 long-ceiling-470102-p4.moulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM long-ceiling-470102-p4.moulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM long-ceiling-470102-p4.moulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15524	1	4	24	440.0	440.0	1
2	16881	1	600	66	432.0	432.0	1
3	16995	1	-1	372	-1.0	-1.0	1
4	18141	1	-12	360	-35.0	-35.0	1
5	13270	1	200	366	590.0	590.0	1
6	16990	1	100	218	179.0	179.0	1
7	17102	1	2	261	26.0	26.0	1
8	13703	1	10	318	100.0	100.0	1
9	16765	1	4	294	34.0	34.0	1
10	15562	1	39	351	135.0	135.0	1
11	13017	1	48	7	204.0	204.0	1

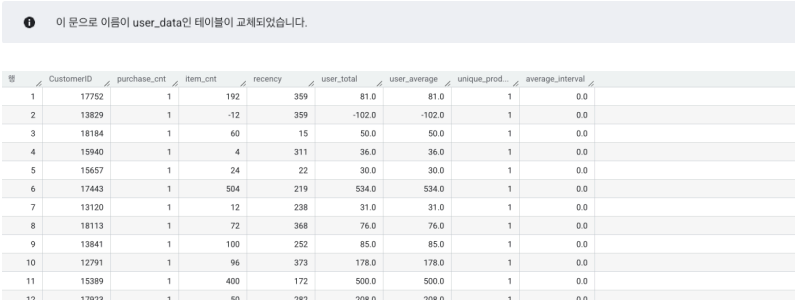
2. 평균 구매 주기

- **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 long-ceiling-470102-p4.moulabs_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
      long-ceiling-470102-p4.moulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM long-ceiling-470102-p4.moulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_interval
1	17752	1	192	359	81.0	81.0	1	0.0
2	13829	1	-12	359	-102.0	-102.0	1	0.0
3	18184	1	60	15	50.0	50.0	1	0.0
4	15940	1	4	311	36.0	36.0	1	0.0
5	15657	1	24	22	30.0	30.0	1	0.0
6	17443	1	504	219	534.0	534.0	1	0.0
7	13120	1	12	238	31.0	31.0	1	0.0
8	18113	1	72	368	76.0	76.0	1	0.0
9	13841	1	100	252	85.0	85.0	1	0.0
10	12791	1	96	373	178.0	178.0	1	0.0
11	15389	1	400	172	500.0	500.0	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 project_name.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    # [[YOUR QUERY]] AS total_transactions,
    # [[YOUR QUERY]] AS cancel_frequency
  FROM project_name.modulabs_project.data
  # [[YOUR QUERY]]
)

SELECT u.*, t.* EXCEPT(CustomerID), # [[YOUR QUERY]] AS cancel_rate
FROM `project_name.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON # [[YOUR QUERY]];
```

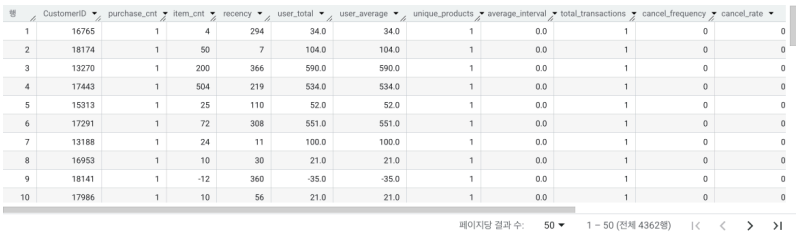
[결과 이미지를 넣어주세요]

이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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

```
SELECT *
FROM long-ceiling-470102-p4.moulabs_project.user_data
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16765	1	4	294	34.0	34.0	1	0.0	1	0	0
2	18174	1	50	7	104.0	104.0	1	0.0	1	0	0
3	13270	1	200	366	590.0	590.0	1	0.0	1	0	0
4	17443	1	504	219	534.0	534.0	1	0.0	1	0	0
5	15313	1	25	110	52.0	52.0	1	0.0	1	0	0
6	17291	1	72	308	551.0	551.0	1	0.0	1	0	0
7	13188	1	24	11	100.0	100.0	1	0.0	1	0	0
8	16953	1	10	30	21.0	21.0	1	0.0	1	0	0
9	18141	1	-12	360	-35.0	-35.0	1	0.0	1	0	0
10	17986	1	10	56	21.0	21.0	1	0.0	1	0	0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

회고

[회고 내용을 작성해주세요]

Keep : 마케팅 측면에서 실무에 가까운 데이터 처리과정을 경험할 수 있어서 재밌었음

Problem :

- MySQL에 익숙하여 BigQuery와의 문법차이 때문에 조금 헥갈렸음.
- project_name.dataset_name.table_name으로 데이터 테이블을 불러오는 방식이 번거로움

Try :

- MySQL과 BigQuery 간의 문법 차이를 비교 정리
- 아래 방법을 활용 예정

1) 기본 프로젝트 / 기본 데이터셋 설정하기

- **쿼리 에디터 상단**에서 "기본 프로젝트"를 지정하면 `project_name`은 생략 가능
- **쿼리 실행 옵션 → 기본 데이터셋(default dataset)** 을 지정하면 `dataset_name`도 생략 가능

```
SELECT *
FROM table_name
```

---

2) 세션 단위 기본 데이터셋 설정 (`SET` 구문)

BigQuery 콘솔 또는 쿼리에서:

```
-- 세션에 기본 데이터셋 설정
SET @@dataset_id = 'dataset_name';
```

→ 이후 쿼리에서는 `dataset_name` 생략 가능:

```
SELECT * FROM table_name;
```

---

3) 뷰(View) 또는 임시 뷰 생성

자주 쓰는 긴 경로를 뷰로 감싸두고 짧게 불러오기:

```
CREATE OR REPLACE VIEW my_dataset.my_table_alias AS
SELECT * FROM `project_name.dataset_name.table_name`;
```

→ 사용 시:

```
SELECT * FROM my_dataset.my_table_alias;
```

---

4) 외부 도구/IDE 활용

- **dbt, Looker, Dataform** 등에서는 프로젝트/데이터셋을 기본값으로 설정 가능
- **Python(BigQuery client)** 에서 `default_dataset` 옵션 지정 가능

정리:

- **빠른 실험/쿼리** → 기본 데이터셋 지정 (쿼리 에디터 설정 or `SET @@dataset_id`)
- **재사용/협업** → View 만들어두고 alias처럼 사용