고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
# [[YOUR QUERY]]
SELECT *
FROM deductive-anvil-456101-v5.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 ...	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-LIGH...	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
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
# [[YOUR QUERY]]
SELECT COUNT(InvoiceNo)
FROM deductive-anvil-456101-v5.modulabs_project.data ;
```

행	f0_
1	541909

데이터 수 세기

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

```
# [[YOUR QUERY]]
SELECT COUNT(InvoiceNo) AS COUNT_InvoiceNo,
COUNT(StockCode) AS COUNT_StockCode,
COUNT(Description) AS Description,
COUNT(Quantity) AS COUNT_Quantity,
COUNT(InvoiceDate) AS InvoiceDate,
COUNT(UnitPrice) AS COUNT_UnitPrice,
COUNT(CustomerID) AS COUNT_Custpmer_ID,
COUNT(Country) AS COUNT_Country
FROM deductive-anvil-456101-v5.modulabs_project.data ;
```

행	COUNT_InvoiceNo	COUNT_StockCode	Description	COUNT_Quantity	InvoiceDate	COUNT_UnitPrice	COUNT_Custpmer_ID	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**

```
# [[YOUR QUERY]]
SELECT "InvoiceNo" AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM deductive-anvil-456101-v5.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 deductive-anvil-456101-v5.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 deductive-anvil-456101-v5.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 deductive-anvil-456101-v5.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 deductive-anvil-456101-v5.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 deductive-anvil-456101-v5.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 deductive-anvil-456101-v5.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 deductive-anvil-456101-v5.modulabs_project.data;
```

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

결측치 처리 전략

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

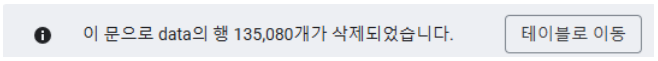
```
SELECT DISTINCT Description
FROM deductive-anvil-456101-v5.modulabs_project.data
WHERE StockCode = '85123A'
ORDER BY Description;
```

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**

```
DELETE FROM deductive-anvil-456101-v5.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**

```
WITH col_group AS (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS count
  FROM deductive-anvil-456101-v5.modulabs_project.data
  GROUP BY
    InvoiceNo, StockCode, Description, Quantity,
    InvoiceDate, UnitPrice, CustomerID, Country
)

SELECT
  COUNT(*) AS same_col
FROM col_group
WHERE count > 1;
```

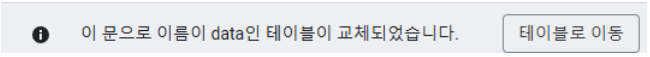
행	same_col
1	4837

중복값 처리

- **중복값을 제거하는 쿼리문 작성하기**

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

```
CREATE OR REPLACE TABLE deductive-anvil-456101-v5.modulabs_project.data AS
SELECT DISTINCT *
FROM deductive-anvil-456101-v5.modulabs_project.data;
```



11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS count_dis_InvoiceNo
FROM deductive-anvil-456101-v5.modulabs_project.data;
```

행	count_dis_InvoiceNo
1	22190

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM deductive-anvil-456101-v5.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

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT *
FROM deductive-anvil-456101-v5.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STOR...	-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	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
6	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
7	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
8	C547388	84050	PINK HEART SHAPE EGG FRYI...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
9	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
10	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
11	C547388	37448	CERAMIC CAKE DESIGN SPOT	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway

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) AS canceled_per
FROM deductive-anvil-456101-v5.modulabs_project.data;
```

행	canceled_per
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS count_dis_StockCode
FROM deductive-anvil-456101-v5.modulabs_project.data;
```

행	count_dis_StockCod
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 deductive-anvil-456101-v5.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DES
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 deductive-anvil-456101-v5.modulabs_project.data
)
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0,1);
```

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)

```
SELECT DISTINCT
  StockCode,
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[A-Za-z]', '')) AS letter_count
FROM (
  SELECT StockCode
  FROM deductive-anvil-456101-v5.modulabs_project.data
)
WHERE
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1);
```

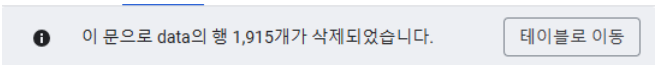
행	StockCode	number_count	letter_count
1	POST	0	4
2	M	0	1
3	C2	1	1
4	D	0	1
5	BANK CHARGES	0	11
6	PADS	0	4
7	DOT	0	3
8	CRUK	0	4

```
SELECT
  ROUND(
    COUNTIF(
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1) ) / COUNT(*) * 100, 2
  ) AS per_str_StockCode
FROM deductive-anvil-456101-v5.modulabs_project.data;
```

행	per_str_StockCode
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

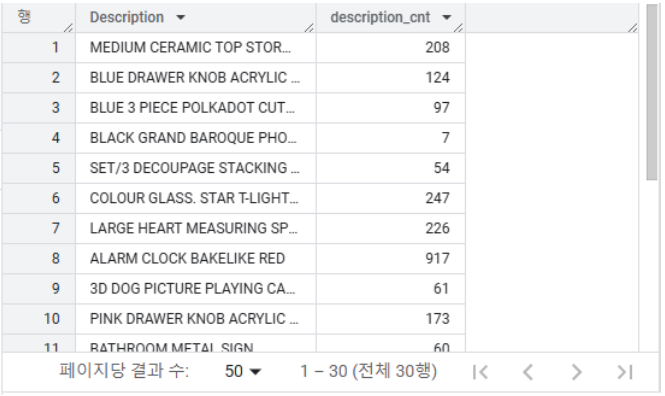
```
DELETE FROM deductive-anvil-456101-v5.modulabs_project.data
WHERE StockCode IN (
 SELECT StockCode
 FROM (
  SELECT StockCode,
       LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM deductive-anvil-456101-v5.modulabs_project.data
 )
 WHERE number_count IN (0, 1)
);
```



Description 살펴보기

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

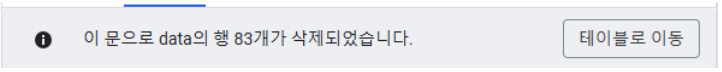
```
SELECT Description, COUNT(*) AS description_cnt
FROM deductive-anvil-456101-v5.modulabs_project.data
GROUP BY Description
LIMIT 30;
```



행	Description	description_cnt
1	MEDIUM CERAMIC TOP STOR...	208
2	BLUE DRAWER KNOB ACRYLIC ...	124
3	BLUE 3 PIECE POLKADOT CUT...	97
4	BLACK GRAND BAROQUE PHO...	7
5	SET/3 DECOUPAGE STACKING ...	54
6	COLOUR GLASS. STAR T-LIGHT...	247
7	LARGE HEART MEASURING SP...	226
8	ALARM CLOCK BAKELIKE RED	917
9	3D DOG PICTURE PLAYING CA...	61
10	PINK DRAWER KNOB ACRYLIC ...	173
11	BATHROOM METAL SIGN	60

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM deductive-anvil-456101-v5.modulabs_project.data
WHERE Description IN ('Next Day Carriage','High Resolution Image');
```

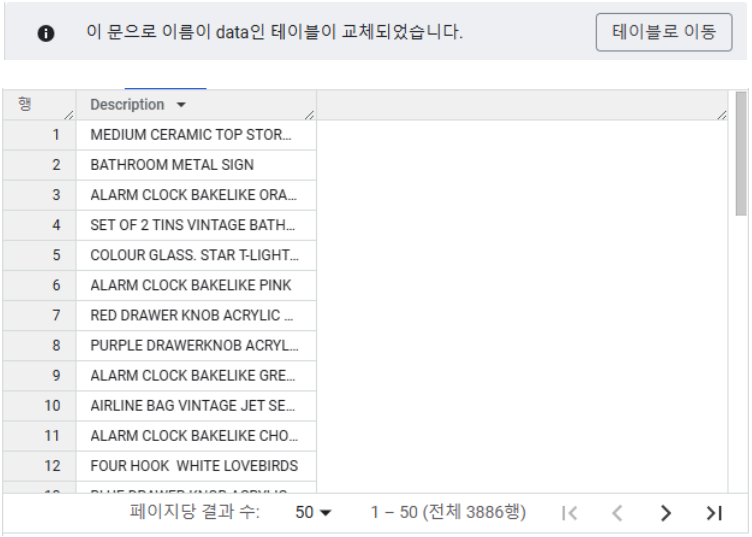


- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE deductive-anvil-456101-v5.modulabs_project.data AS
SELECT
 * EXCEPT (Description),
```

```
  UPPER(Description) AS Description
FROM deductive-anvil-456101-v5.modulabs_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 deductive-anvil-456101-v5.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**

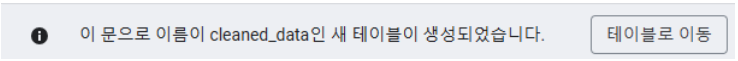
```
SELECT
  COUNT(Quantity) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM deductive-anvil-456101-v5.modulabs_project.data;
```

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	399606	-80995	80995	12.23171824246...

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

```
CREATE OR REPLACE TABLE deductive-anvil-456101-v5.modulabs_project.cleaned_data AS
SELECT *
```

```
FROM deductive-anvil-456101-v5.modulabs_project.data
WHERE UnitPrice != 0;
```



11-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay
FROM deductive-anvil-456101-v5.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

페이지당 결과 수: 50 1 – 50 (전체 399606행)

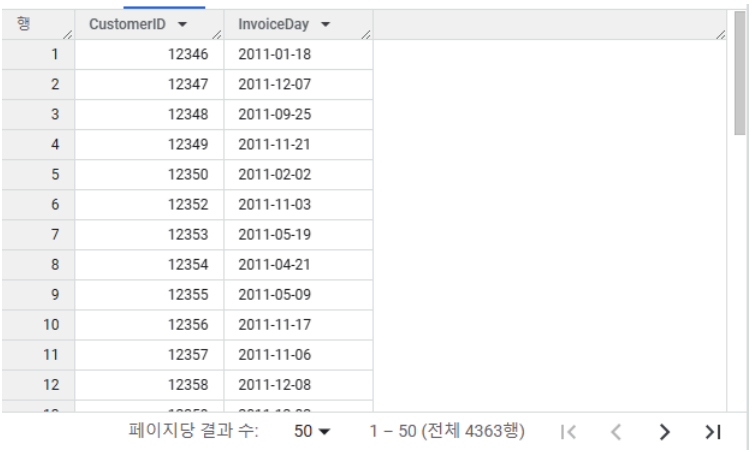
- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
MAX(InvoiceDate) AS most_recent_date,
DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM deductive-anvil-456101-v5.modulabs_project.data;
```

행	most_recent_date	InvoiceDay
1	2011-12-09 12:50:00 UTC	2011-12-09

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

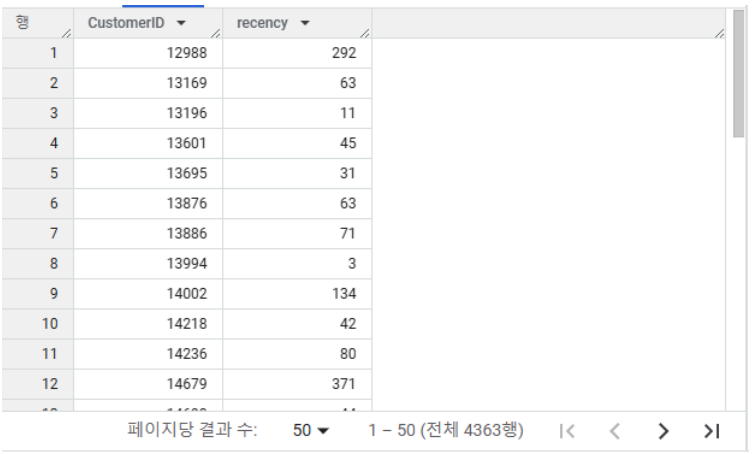
```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM deductive-anvil-456101-v5.modulabs_project.data
GROUP BY CustomerID;
```



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

- **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 deductive-anvil-456101-v5.modulabs_project.data
  GROUP BY CustomerID
);
```

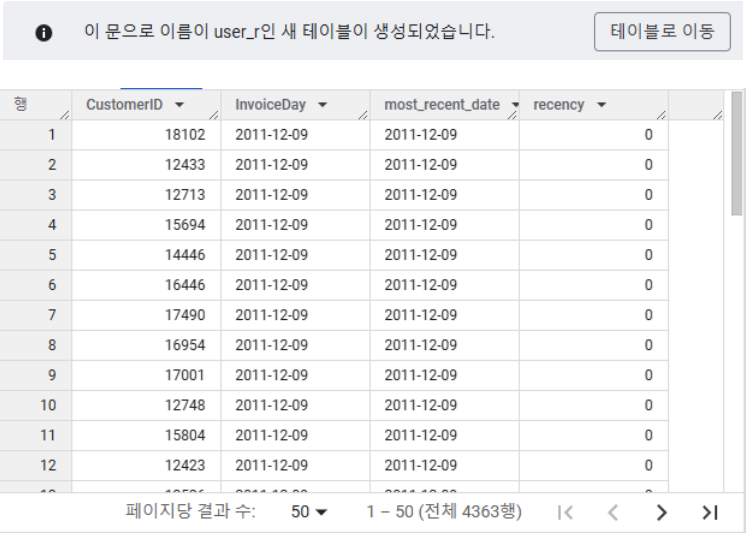


행	CustomerID	recency
1	12988	292
2	13169	63
3	13196	11
4	13601	45
5	13695	31
6	13876	63
7	13886	71
8	13994	3
9	14002	134
10	14218	42
11	14236	80
12	14679	371

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

```
CREATE OR REPLACE TABLE deductive-anvil-456101-v5.modulabs_project.user_r AS
WITH Invoice AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM deductive-anvil-456101-v5.modulabs_project.data
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
```

```
SELECT
  CustomerID,
  InvoiceDay,
  MAX(InvoiceDay) OVER () AS most_recent_date,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM Invoice;
```



행	CustomerID	InvoiceDay	most_recent_date	recency
1	18102	2011-12-09	2011-12-09	0
2	12433	2011-12-09	2011-12-09	0
3	12713	2011-12-09	2011-12-09	0
4	15694	2011-12-09	2011-12-09	0
5	14446	2011-12-09	2011-12-09	0
6	16446	2011-12-09	2011-12-09	0
7	17490	2011-12-09	2011-12-09	0
8	16954	2011-12-09	2011-12-09	0
9	17001	2011-12-09	2011-12-09	0
10	12748	2011-12-09	2011-12-09	0
11	15804	2011-12-09	2011-12-09	0
12	12423	2011-12-09	2011-12-09	0

페이지당 결과 수: 50 1 – 50 (전체 4363행)

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM deductive-anvil-456101-v5.modulabs_project.data
GROUP BY CustomerID;
```

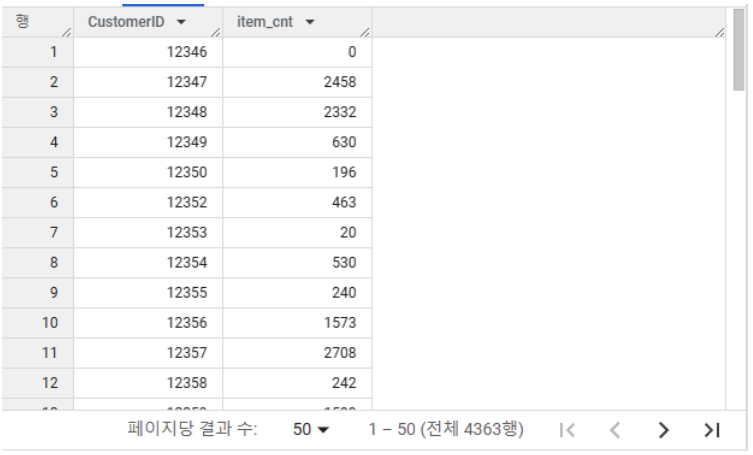
행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1
8	12354	1
9	12355	1
10	12356	3
11	12357	1
12	12358	2

페이지당 결과 수: 50 1 – 50 (전체 4363행)

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
```

```
  SUM(Quantity) AS item_cnt
FROM deductive-anvil-456101-v5.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
10	12356	1573
11	12357	2708
12	12358	242

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

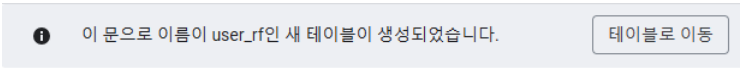
```
CREATE OR REPLACE TABLE deductive-anvil-456101-v5.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH
purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM
    deductive-anvil-456101-v5.modulabs_project.data
  GROUP BY
    CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM
    deductive-anvil-456101-v5.modulabs_project.data
  GROUP BY
    CustomerID
)

-- (3) 기존의 user_r과 JOIN
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
```

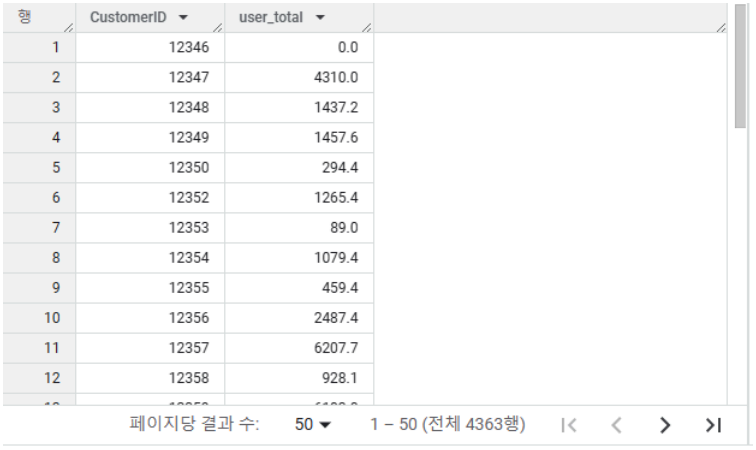
```
deductive-anvil-456101-v5.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```



Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

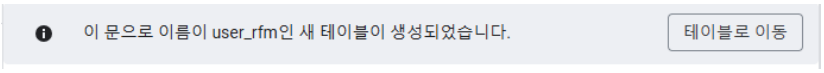
```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM deductive-anvil-456101-v5.modulabs_project.data
GROUP BY CustomerID;
```



행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.6
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7
12	12358	928.1

- **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**

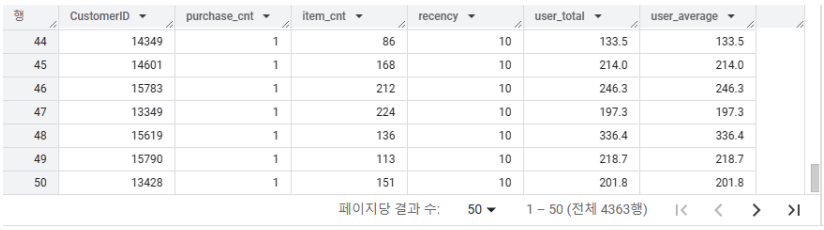
```
CREATE OR REPLACE TABLE deductive-anvil-456101-v5.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(ut.user_total / rf.purchase_cnt, 1) AS user_average
FROM deductive-anvil-456101-v5.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM deductive-anvil-456101-v5.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM deductive-anvil-456101-v5.modulabs_project.user_rfm ;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
44	14349	1	86	10	133.5	133.5
45	14601	1	168	10	214.0	214.0
46	15783	1	212	10	246.3	246.3
47	13349	1	224	10	197.3	197.3
48	15619	1	136	10	336.4	336.4
49	15790	1	113	10	218.7	218.7
50	13428	1	151	10	201.8	201.8

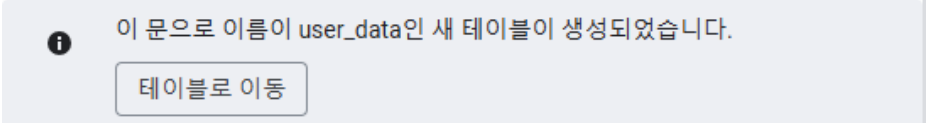
페이지당 결과 수: 50 1 – 50 (전체 4363행)

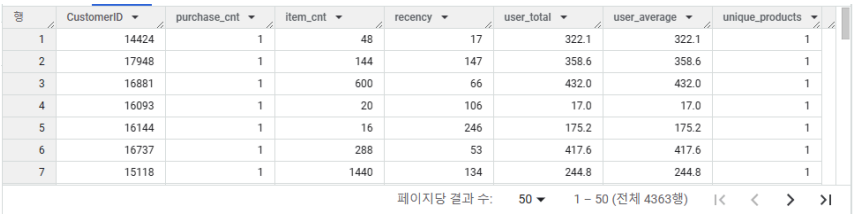
11-8. 추가 Feature 추출

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 deductive-anvil-456101-v5.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM deductive-anvil-456101-v5.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM deductive-anvil-456101-v5.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	14424	1	48	17	322.1	322.1	1
2	17948	1	144	147	358.6	358.6	1
3	16881	1	600	66	432.0	432.0	1
4	16093	1	20	106	17.0	17.0	1
5	16144	1	16	246	175.2	175.2	1
6	16737	1	288	53	417.6	417.6	1
7	15118	1	1440	134	244.8	244.8	1

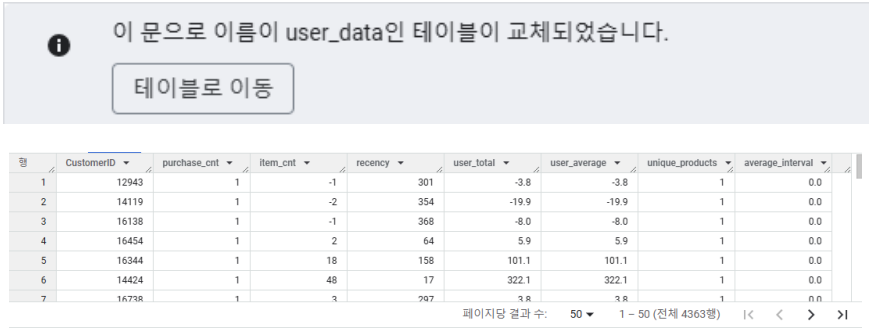
페이지당 결과 수: 50 1 – 50 (전체 4363행)

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 deductive-anvil-456101-v5.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
      deductive-anvil-456101-v5.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM deductive-anvil-456101-v5.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
1	12943	1	-1	301	-3.8	-3.8	1	0.0
2	14119	1	-2	354	-19.9	-19.9	1	0.0
3	16138	1	-1	368	-8.0	-8.0	1	0.0
4	16454	1	2	64	5.9	5.9	1	0.0
5	16344	1	18	158	101.1	101.1	1	0.0
6	14424	1	48	17	322.1	322.1	1	0.0
7	16738	1	3	297	3.8	3.8	1	0.0

페이지당 결과 수: 50 1 – 50 (전체 4363행)

3. 구매 취소 경향성

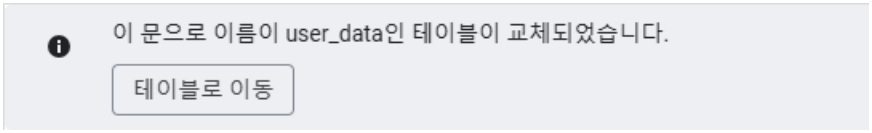
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE deductive-anvil-456101-v5.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT InvoiceNo LIKE 'C%') AS cancel_frequency
  FROM deductive-anvil-456101-v5.modulabs_project.data
```

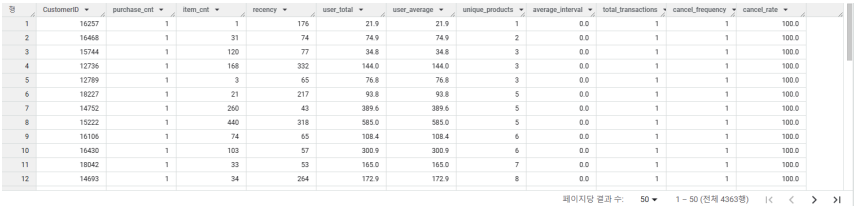
```
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.total_transactions,
  t.cancel_frequency,
  ROUND(t.cancel_frequency / t.total_transactions * 100, 2) AS cancel_rate
FROM
  deductive-anvil-456101-v5.modulabs_project.user_data AS u
LEFT JOIN
  TransactionInfo AS t
ON
  u.CustomerID = t.CustomerID;
```



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

```
SELECT *
FROM deductive-anvil-456101-v5.modulabs_project.user_data;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16257	1	1	176	21.9	21.9	1	0.0	1	1	100.0
2	16468	1	31	74	74.9	74.9	2	0.0	1	1	100.0
3	15744	1	120	77	34.8	34.8	3	0.0	1	1	100.0
4	12736	1	168	332	144.0	144.0	3	0.0	1	1	100.0
5	12789	1	3	65	76.8	76.8	3	0.0	1	1	100.0
6	18227	1	21	217	93.8	93.8	5	0.0	1	1	100.0
7	14752	1	260	43	389.6	389.6	5	0.0	1	1	100.0
8	15222	1	440	318	585.0	585.0	5	0.0	1	1	100.0
9	16106	1	74	65	108.4	108.4	6	0.0	1	1	100.0
10	16430	1	103	57	300.9	300.9	6	0.0	1	1	100.0
11	18042	1	33	53	165.0	165.0	7	0.0	1	1	100.0
12	14693	1	34	264	172.9	172.9	8	0.0	1	1	100.0

회고

[회고 내용을 작성해주세요]

Keep : sql과 관련한 다양한 데이터 분석 및 테이블 생성과 관련하여 다양한 실습을 할 수 있었음

Problem : 중간에 모르거나 헤깔리는 부분이 있어 막히는 구간이 있었는데 학습이 더 필요할 것 같음

Try : 다양한 데이터를 활용하여 더욱 많은 방법으로 분석을 해보고 싶다. 다양한 기법들을 사용해보고 싶다. 프로그래밍에 정답은 없고 한 가지를 하기 위하여 다양한 방법으로 프로그래밍을 할 수 있으므로 여러가지를 해보며 더욱 효율적인 방법을 찾아보겠다.