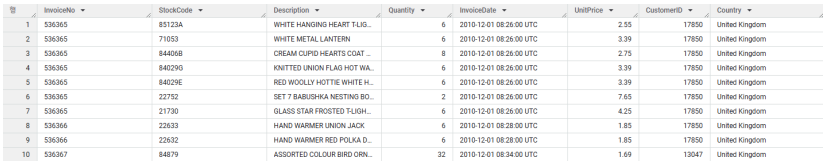
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

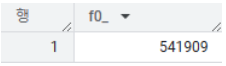
```
SELECT *
FROM quiet-spirit-456102-s9.modulabs_project.data
LIMIT 10 ;
```



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**

```
SELECT Count(InvoiceDate)
FROM quiet-spirit-456102-s9.modulabs_project.data ;
```

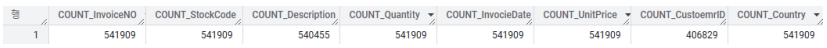


행	f0_
1	541909

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_InvocieDate,
COUNT(UnitPrice) AS COUNT_UnitPrice,
COUNT(CustomerID) AS COUNT_CustoemrID,
COUNT(Country) AS COUNT_Country
FROM quiet-spirit-456102-s9.modulabs_project.data ;
```



행	COUNT_InvoiceNO	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvocieDate	COUNT_UnitPrice	COUNT_CustoemrID	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**

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

```
FROM quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.modulabs_project.data ;
```

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

결측치 처리 전략

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

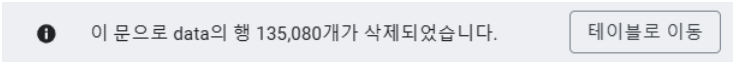
```
SELECT DISTINCT(Description)
FROM quiet-spirit-456102-s9.modulabs_project.data
WHERE StockCode = '85123A' ;
```

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

결측치 처리

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

```
DELETE FROM quiet-spirit-456102-s9.modulabs_project.data
WHERE Description IS NULL
OR CustomerID 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 *
FROM quiet-spirit-456102-s9.modulabs_project.data
GROUP BY InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
HAVING COUNT(*) > 1 ;
```

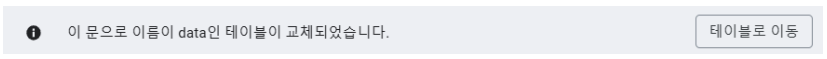
행	InvoiceNo	StockCode	Description	Quantity
1	571034	23239	SET OF 4 KNICK KNACK TINS ...	6
2	571034	23245	SET OF 3 REGENCY CAKE TINS	4
3	571034	23494	VINTAGE DOILY DELUXE SEWI...	3
4	538826	22749	FELTCRAFT PRINCESS CHARL...	1
5	577228	84580	MOUSE TOY WITH PINK T-SHIRT	1
6	577228	23048	SET OF 10 LANTERNS FAIRY LI...	1
7	577228	22435	SET OF 9 HEART SHAPED BAL...	1
8	577228	22270	HAPPY EASTER HANGING DEC...	1
9	577228	22144	CHRISTMAS CRAFT LITTLE FRI...	1
10	577228	23156	SET OF 5 MINI GROCERY MAG...	1
11	539419	48138	DOORMAT UNION FLAG	10
12	538174	22326	ROUND SNACK BOXES SET OF...	12
13	555162	22704	WRAP RED APPLES	25

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 quiet-spirit-456102-s9.modulabs_project.data AS
SELECT DISTINCT *
FROM quiet-spirit-456102-s9.modulabs_project.data ;
```



행	f0_
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM quiet-spirit-456102-s9.modulabs_project.data ;
```

행	f0_
1	22190

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM quiet-spirit-456102-s9.modulabs_project.data
LIMIT 100 ;
```

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

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT *
FROM quiet-spirit-456102-s9.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

행	InvoiceNo	StockCode	Description	Quantity
1	C541433	23166	MEDIUM CERAMIC TOP STOR...	-74215
2	C545329	M	Manual	-1
3	C545329	M	Manual	-1
4	C545330	M	Manual	-1
5	C547388	22701	PINK DOG BOWL	-6
6	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12
7	C547388	21914	BLUE HARMONICA IN BOX	-12
8	C547388	22784	LANTERN CREAM GAZEBO	-3
9	C547388	84050	PINK HEART SHAPE EGG FRYI...	-12
10	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12
11	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6
12	C549955	22839	3 TIER CAKE TIN GREEN AND ...	-2
13	C549955	22666	RECIPE BOX PANTRY YELLOW ...	-2

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(InvoiceNo)*100, 1)
FROM quiet-spirit-456102-s9.modulabs_project.data;
```

행	f0_
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode)
FROM quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

행	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
9	22197	1110
10	23203	1108

페이지당 결과 수: 50 1 – 10 (전체 10행)

- `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 quiet-spirit-456102-s9.modulabs_project.data
)
WHERE number_count between 0 and 1
```

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

- `StockCode`**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
SELECT ROUND(
  SUM(CASE WHEN StockCode IN (
    SELECT StockCode
    FROM quiet-spirit-456102-s9.modulabs_project.data
    WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) BETWEEN 0 AND 1) THEN 1   ELSE 0
    END) / COUNT(StockCode)*100, 2)
FROM quiet-spirit-456102-s9.modulabs_project.data;
```

행	f0_
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM quiet-spirit-456102-s9.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM quiet-spirit-456102-s9.modulabs_project.data
    WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) BETWEEN 0 AND 1)
);
```

이 문으로 data의 행 1,915개가 삭제되었습니다. 테이블로 이동

Description 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM quiet-spirit-456102-s9.modulabs_project.data
GROUP BY Description
ORDER BY 2 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 ...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE...	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000
14	HEART OF WICKER SMALL	990

페이지당 결과 수: 50 1 – 30 (전체 30행)

- **서비스 관련 정보를 포함하는 행들을 제거하기**

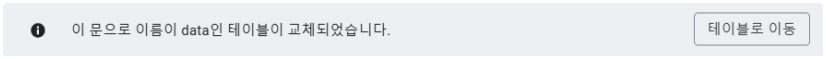
```
DELETE
FROM quiet-spirit-456102-s9.modulabs_project.data
WHERE Description = 'Next Day Carriage'
OR Description = 'High Resolution Image';
```

이 문으로 data의 행 83개가 삭제되었습니다. 테이블로 이동

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE quiet-spirit-456102-s9.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
```

```
 UPPER(Description) AS Description
FROM quiet-spirit-456102-s9.modulabs_project.data;
```



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM quiet-spirit-456102-s9.modulabs_project.data;
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

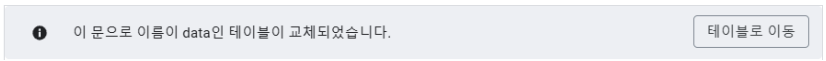
- **단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기**

```
SELECT COUNT(Quantity) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS
FROM quiet-spirit-456102-s9.modulabs_project.data
WHERE UnitPrice = 0;
```

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

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

```
CREATE OR REPLACE TABLE quiet-spirit-456102-s9.modulabs_project.data AS
SELECT *
FROM quiet-spirit-456102-s9.modulabs_project.data
WHERE UnitPrice NOT IN (0);
```



11-7. RFM 스코어

Recency

- **InvoiceDate 컬럼을 연월일 자료형으로 변경하기**

```
SELECT DATE(TIMESTAMP (InvoiceDate)) AS InvoiceDay, *
FROM quiet-spirit-456102-s9.modulabs_project.data;
```

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
20	2010-12-07	537626	22494	12	2010-12-07 1
21	2010-12-07	537626	22195	12	2010-12-07 1
22	2010-12-07	537626	22772	12	2010-12-07 1
23	2010-12-07	537626	84969	6	2010-12-07 1
24	2010-12-07	537626	85116	12	2010-12-07 1
25	2010-12-07	537626	22728	4	2010-12-07 1
26	2010-12-07	537626	22492	36	2010-12-07 1
27	2010-12-07	537626	84997C	6	2010-12-07 1
28	2010-12-07	537626	22725	4	2010-12-07 1
29	2010-12-07	537626	22375	4	2010-12-07 1
30	2010-12-07	537626	85167B	30	2010-12-07 1
31	2010-12-07	537626	22771	12	2010-12-07 1
32	2010-12-07	537626	21171	12	2010-12-07 1

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT  DATE(TIMESTAMP(FIRST_VALUE(InvoiceDate) OVER(ORDER BY InvoiceDate DESC))) AS most_recent_date,
  DATE(TIMESTAMP(InvoiceDate)) AS InvoiceDay, *
FROM quiet-spirit-456102-s9.modulabs_project.data;
```

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity
1	2011-12-09	2011-12-09	581587	22629	
2	2011-12-09	2011-12-09	581587	22631	
3	2011-12-09	2011-12-09	581587	23256	
4	2011-12-09	2011-12-09	581587	22367	
5	2011-12-09	2011-12-09	581587	22727	
6	2011-12-09	2011-12-09	581587	22556	
7	2011-12-09	2011-12-09	581587	22138	
8	2011-12-09	2011-12-09	581587	22555	
9	2011-12-09	2011-12-09	581587	22613	
10	2011-12-09	2011-12-09	581587	22726	
11	2011-12-09	2011-12-09	581587	23255	
12	2011-12-09	2011-12-09	581587	22728	
13	2011-12-09	2011-12-09	581587	22899	

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

```
SELECT
  CustomerID,
  DATE(TIMESTAMP(MAX(InvoiceDate))) AS InvoiceDay
FROM quiet-spirit-456102-s9.modulabs_project.data
GROUP BY CustomerID;
```

행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17
11	12357	2011-11-06
12	12358	2011-12-08
13	12359	2011-12-02
14	12360	2011-10-18

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

```
WITH EX_1 AS (
  SELECT DATE(TIMESTAMP(FIRST_VALUE(InvoiceDate) OVER(ORDER BY InvoiceDate DESC))) AS  most_recent_date,
  DATE(TIMESTAMP(FIRST_VALUE(InvoiceDate) OVER(PARTITION BY CustomerID ORDER BY InvoiceDate DESC))) AS  InvoiceDa
FROM quiet-spirit-456102-s9.modulabs_project.data
)

SELECT CustomerID, most_recent_date, InvoiceDay,
  DATE_DIFF(most_recent_date,InvoiceDay,DAY)
FROM EX_1
GROUP BY CustomerID, most_recent_date, InvoiceDay;
```

행	CustomerID	recency
1	12346	325
2	12347	2
3	12348	75
4	12349	18
5	12350	310
6	12352	36
7	12353	204
8	12354	232
9	12355	214
10	12356	22
11	12357	33
12	12358	1
13	12359	7
14	12360	52

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

```
CREATE OR REPLACE TABLE quiet-spirit-456102-s9.modulabs_project.user_r AS
WITH EX_1 AS (
  SELECT DATE(TIMESTAMP(FIRST_VALUE(InvoiceDate) OVER(ORDER BY InvoiceDate DESC))) AS  most_recent_date,
  DATE(TIMESTAMP(FIRST_VALUE(InvoiceDate) OVER(PARTITION BY CustomerID ORDER BY InvoiceDate DESC))) AS  InvoiceDa
FROM quiet-spirit-456102-s9.modulabs_project.data
)

SELECT CustomerID, most_recent_date, InvoiceDay,
  DATE_DIFF(most_recent_date,InvoiceDay,DAY)
FROM EX_1
GROUP BY CustomerID, most_recent_date, InvoiceDay;
```

이 문으로 이름이 user_r인 테이블이 교체되었습니다. 테이블로 이동

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM quiet-spirit-456102-s9.modulabs_project.data
GROUP BY CustomerID;
```

행	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84
7	12353	4
8	12354	58
9	12355	13
10	12356	58
11	12357	131
12	12358	17
13	12359	251
14	12360	126

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM quiet-spirit-456102-s9.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
13	12359	1599
14	12360	1156

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

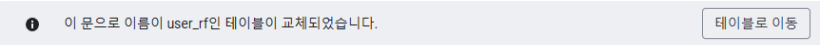
```
CREATE OR REPLACE TABLE quiet-spirit-456102-s9.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM quiet-spirit-456102-s9.modulabs_project.data
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
```

```
item_cnt AS (
  SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
  FROM quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```



Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice),1) AS user_total
FROM quiet-spirit-456102-s9.modulabs_project.data
GROUP BY CustomerID;
```

행	CustomerID	user_total
1	12346	2.1
2	12347	481.2
3	12348	18.7
4	12349	305.1
5	12350	25.3
6	12352	330.5
7	12353	24.3
8	12354	261.2
9	12355	54.6
10	12356	170.9
11	12357	438.7
12	12358	77.2
13	12359	2210.1
14	12360	337.9

- **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 quiet-spirit-456102-s9.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
```

```
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(user_total / rf.purchase_cnt ,1)  AS user_average
FROM quiet-spirit-456102-s9.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice*Quantity),0) AS user_total
    FROM quiet-spirit-456102-s9.modulabs_project.data
    GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다. 테이블로 이동

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM quiet-spirit-456102-s9.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	13436	1	76	1	197.0	197.0
3	14569	1	79	1	227.0	227.0
4	15520	1	314	1	344.0	344.0
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	14578	1	240	3	169.0	169.0
10	17914	1	457	3	329.0	329.0
11	12478	1	233	3	546.0	546.0
12	16569	1	93	3	124.0	124.0
13	12650	1	250	3	242.0	242.0
14	16528	1	171	3	244.0	244.0
15	15992	1	17	3	42.0	42.0
16	12442	1	181	3	144.0	144.0

11-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

- **1) 고객 별로 구매한 상품들의 고유한 수를 계산하기**
  2)
  `user_rfm` **테이블과 결과를 합치기**
  3)
  `user_data` **라는 이름의 테이블에 저장하기**

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
```

```
  COUNT(DISTINCT StockCode) AS unique_products
 FROM project_name.modulabs_project.data
 GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다. 테이블로 이동

2. 평균 구매 주기

- **고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)**
  - **균 구매 소요 일수를 계산하고, 그 결과를 `user_data` 에 통합**

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS
WITH purchase_intervals AS (
 -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
 SELECT
  CustomerID,
  CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
 FROM (
  -- (1) 구매와 구매 사이에 소요된 일수
  SELECT
   CustomerID,
   DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
  FROM
   project_name.modulabs_project.data
  WHERE CustomerID IS NOT NULL
 )
 GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

이 문으로 이름이 user_data인 테이블이 교체되었습니다. 테이블로 이동

3. 구매 취소 경향성

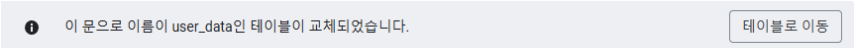
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE quiet-spirit-456102-s9.modulabs_project.user_data AS

WITH TransactionInfo AS (
 SELECT
  CustomerID,
```

```
  COUNT(InvoiceNo) AS total_transactions,
  SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
 FROM quiet-spirit-456102-s9.modulabs_project.data
 GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_transactions*100, 2) AS cancel_rate
FROM quiet-spirit-456102-s9.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 quiet-spirit-456102-s9.modulabs_project.user_data;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	uniqu
1	13099	1	288	99	207.0	207.0	
2	16061	1	-1	269	-30.0	-30.0	
3	16995	1	-1	372	-1.0	-1.0	
4	15195	1	1404	2	3861.0	3861.0	
5	18113	1	72	368	76.0	76.0	
6	15510	1	2	330	250.0	250.0	
7	16078	1	16	283	79.0	79.0	
8	18068	1	6	289	102.0	102.0	
9	15070	1	36	372	106.0	106.0	
10	17443	1	504	219	534.0	534.0	
11	14090	1	72	324	76.0	76.0	
12	16579	1	-12	365	-31.0	-31.0	
13	17752	1	192	359	81.0	81.0	
14	13135	1	4300	196	3096.0	3096.0	
15	17763	1	12	263	15.0	15.0	

회고

[회고 내용을 작성해주세요]

Keep : 생각보다 문제가 잘 이해되었습니다.

평소에 끝나고도 공부를 한게 도움이 된것같습니다.

Problem : 오류값처리가 어려워서 한참을 헤맸습니다.

그리고 마음이 급해서 출력 결과도 노션에 붙여 넣지 않고,

다음 테스트를 진행해서 쿼리문을 처음부터 다시 실행시킨 경우가 있어 힘들었습니다.

Try : 다음번에는 좀 더 천천히 꼼꼼하게 하겠습니다.