고객을 세그먼테이션하자! [프로젝트] - 정다빈

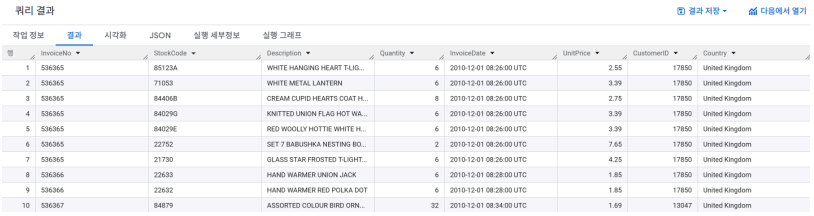
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT * ,
FROM symbolic-folio-470301-r6.modulabs_project.data
LIMIT 10
```

[결과 이미지를 넣어주세요]

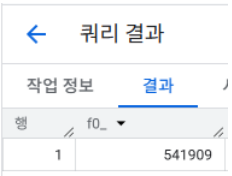


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	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(*)
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]



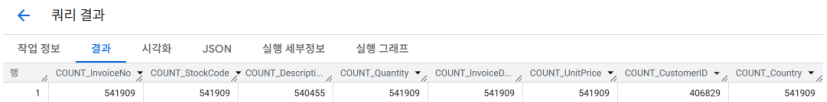
행	f0_
1	541909

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 symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]



행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	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**

```
SELECT
   'InvoiceNo' AS column_name,
   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE StockCode = '85123A'
ORDER BY Description
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON

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

결측치 처리

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

```
DELETE FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE Description is NULL

DELETE FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE CustomerID is NULL
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 data의 행 1,454개가 삭제되었습니다.

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 data의 행 133,626개가 삭제되었습니다.

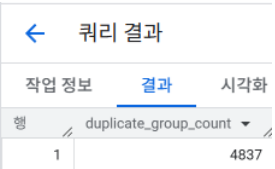
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
WITH counttable AS (
  SELECT COUNT(*) AS dup_count
  FROM symbolic-folio-470301-r6.modulabs_project.data
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
)
SELECT SUM(1) AS duplicate_group_count
FROM counttable
```

[결과 이미지를 넣어주세요]

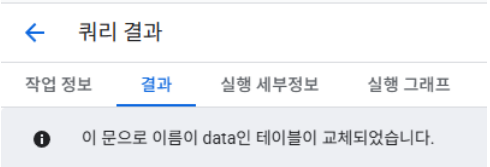


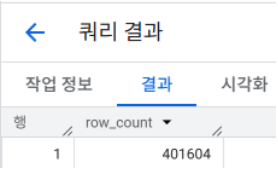
중복값 처리

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

```
CREATE OR REPLACE TABLE symbolic-folio-470301-r6.modulabs_project.data AS
SELECT DISTINCT *
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]





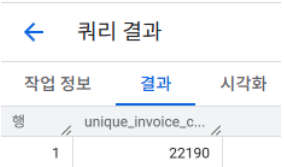
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM symbolic-folio-470301-r6.modulabs_project.data
```

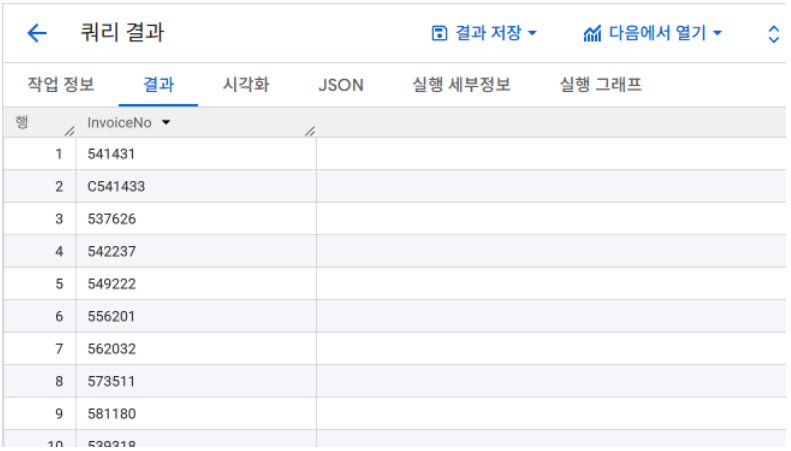
[결과 이미지를 넣어주세요]



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

```
SELECT DISTINCT InvoiceNo
FROM symbolic-folio-470301-r6.modulabs_project.data
LIMIT 100
```

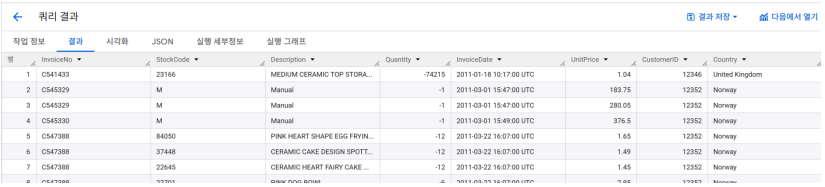
[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100
```

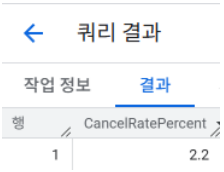
[결과 이미지를 넣어주세요]



- **구매 건 상태가 `Canceled` 인 데이터의 비율(%) - 소수점 첫번째 자리까지**

```
SELECT ROUND(
  SUM(CASE WHEN STARTS_WITH(InvoiceNo, 'C') THEN 1 ELSE 0 END) / COUNT(*) * 100, 1
) AS CancelRatePercent
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

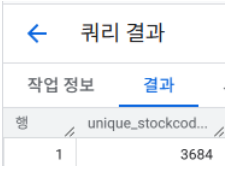


StockCode 살펴보기

- **고유한 StockCode 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]



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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM symbolic-folio-470301-r6.modulabs_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 symbolic-folio-470301-r6.modulabs_project.data
)
WHERE number_count <= 1
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세

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(
    COUNTIF(
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1
    ) / COUNT(*) * 100
  , 2) AS percentage
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과

행	percentage
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM symbolic-folio-470301-r6.modulabs_project.data
  WHERE NOT REGEXP_CONTAINS(StockCode, r'[0-9]')
  OR StockCode IN ('POST', 'BANK CHARGES', 'DOT', 'CRUK', 'PADS', 'M', 'D', 'C2')
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

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 symbolic-folio-470301-r6.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세

행	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY D...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE ...	1062
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
15	NATURAL SLATE HEART CHAL...	989
16	JAM MAKING SET WITH JARS	966
17	LUNCH BAG PINK POLKADOT	961
18	LUNCH BAG SUKI DESIGN	932
19	ALARM CLOCK BAKELIKE RED	917
20	WOODEN PICTURE FRAME WHI...	900
21	REX CASH+CARRY JUMBO SHO...	900
22	JUMBO BAG PINK POLKADOT	897
23	LUNCH BAG APPLE DESIGN	890
24	SET OF 4 PANTRY JELLY MOUL...	890
25	BAKING SET 9 PIECE RETROSP...	885
26	RECIPE BOX PANTRY YELLOW ...	883
27	JAM MAKING SET PRINTED	883
28	LUNCH BAG WOODLAND	850
29	ROSES REGENCY TEACUP AND ...	844
30	VICTORIAN GLASS HANGING T-...	843

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE LOWER(Description) LIKE '%next day carriage%'
  OR LOWER(Description) LIKE '%high resolution image%';
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

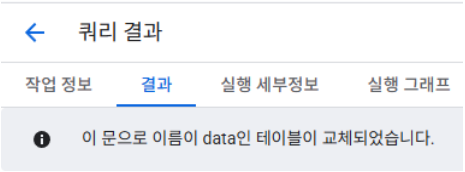
이 문으로 data의 행 83개가 삭제되었습니다.

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE symbolic-folio-470301-r6.modulabs_project.data AS
SELECT
```

```
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

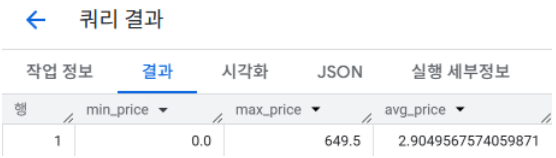


UnitPrice 살펴보기

- **UnitPrice 의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

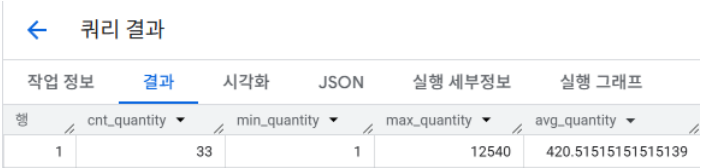


행	min_price	max_price	avg_price
1	0.0	649.5	2.9049567574059871

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

```
SELECT
  COUNT(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

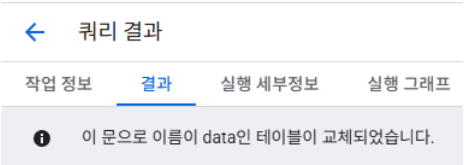


행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.51515151515139

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

```
CREATE OR REPLACE TABLE symbolic-folio-470301-r6.modulabs_project.data AS
SELECT *
FROM symbolic-folio-470301-r6.modulabs_project.data
WHERE UnitPrice > 0
```

[결과 이미지를 넣어주세요]



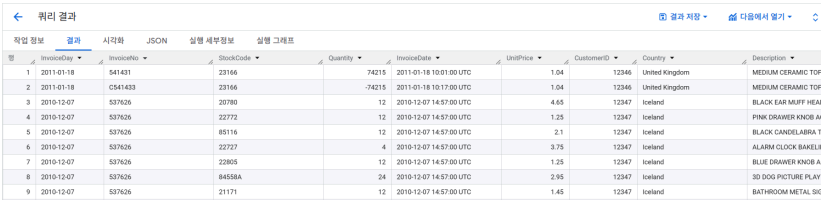
11-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

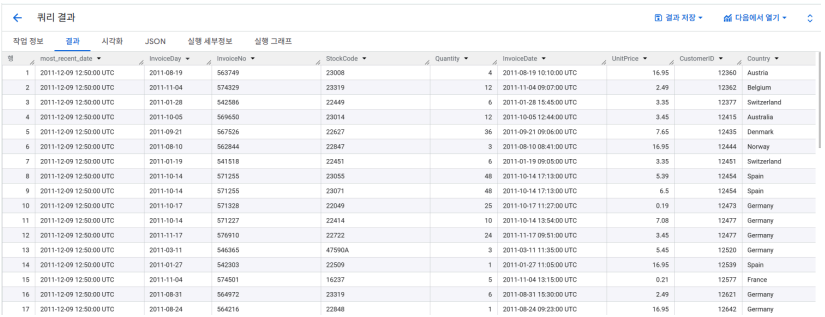


행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOF
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOF
3	2010-12-07	537626	20780	12	2010-12-07 14:57:00 UTC	4.65	12347	Iceland	BLACK EAR MUFF HEA
4	2010-12-07	537626	22772	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	PINK DRAWER KNOB A
5	2010-12-07	537626	85116	12	2010-12-07 14:57:00 UTC	2.1	12347	Iceland	BLACK CANDELABRA T
6	2010-12-07	537626	22727	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELI
7	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB A
8	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC	2.95	12347	Iceland	3D DOG PICTURE PLAY
9	2010-12-07	537626	21171	12	2010-12-07 14:57:00 UTC	1.45	12347	Iceland	BATHROOM METAL SIG

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(InvoiceDate) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM symbolic-folio-470301-r6.modulabs_project.data
```

[결과 이미지를 넣어주세요]



행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-12-09 12:50:00 UTC	2011-08-19	563749	23008	4	2011-08-19 10:10:00 UTC	16.95	12360	Austria
2	2011-12-09 12:50:00 UTC	2011-11-04	574329	23319	12	2011-11-04 09:07:00 UTC	2.49	12362	Belgium
3	2011-12-09 12:50:00 UTC	2011-01-28	542586	22449	6	2011-01-28 15:45:00 UTC	3.35	12377	Switzerland
4	2011-12-09 12:50:00 UTC	2011-10-05	569650	23014	12	2011-10-05 12:44:00 UTC	3.45	12415	Australia
5	2011-12-09 12:50:00 UTC	2011-09-21	567526	22627	36	2011-09-21 09:06:00 UTC	7.65	12435	Denmark
6	2011-12-09 12:50:00 UTC	2011-08-10	562844	22847	3	2011-08-10 08:41:00 UTC	16.95	12444	Norway
7	2011-12-09 12:50:00 UTC	2011-01-19	541518	22451	6	2011-01-19 09:05:00 UTC	3.35	12451	Switzerland
8	2011-12-09 12:50:00 UTC	2011-10-14	571255	23055	48	2011-10-14 17:13:00 UTC	5.39	12454	Spain
9	2011-12-09 12:50:00 UTC	2011-10-14	571255	23071	48	2011-10-14 17:13:00 UTC	6.5	12454	Spain
10	2011-12-09 12:50:00 UTC	2011-10-17	571328	22049	25	2011-10-17 11:27:00 UTC	0.19	12473	Germany
11	2011-12-09 12:50:00 UTC	2011-10-14	571227	22414	10	2011-10-14 13:54:00 UTC	7.08	12477	Germany
12	2011-12-09 12:50:00 UTC	2011-11-17	576910	22722	24	2011-11-17 09:51:00 UTC	3.45	12477	Germany
13	2011-12-09 12:50:00 UTC	2011-03-11	546365	47590A	3	2011-03-11 11:35:00 UTC	5.45	12520	Germany
14	2011-12-09 12:50:00 UTC	2011-01-27	542303	22509	1	2011-01-27 11:05:00 UTC	16.95	12539	Spain
15	2011-12-09 12:50:00 UTC	2011-11-04	574501	16237	5	2011-11-04 13:15:00 UTC	0.21	12577	France
16	2011-12-09 12:50:00 UTC	2011-08-31	564972	23319	6	2011-08-31 15:30:00 UTC	2.49	12621	Germany
17	2011-12-09 12:50:00 UTC	2011-08-24	564216	22848	1	2011-08-24 09:23:00 UTC	16.95	12642	Germany

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

```
SELECT
   CustomerID,
   MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM symbolic-folio-470301-r6.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON

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
15	12361	2011-02-25
16	12362	2011-12-06

- **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
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	CustomerID	recency
1	12559	310
2	12738	372
3	12842	70
4	13334	82
5	13350	16
6	13355	122
7	13560	7
8	13579	14
9	13650	16
10	13707	267
11	13741	71
12	14060	4
13	14317	64
14	14733	9
15	14741	11
16	14755	9
17	14803	162
18	14821	366

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

```
CREATE OR REPLACE TABLE symbolic-folio-470301-r6.modulabs_project.user_r AS

WITH user_last_purchase AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM symbolic-folio-470301-r6.modulabs_project.data
  GROUP BY CustomerID
),

most_recent AS (
  SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
  FROM symbolic-folio-470301-r6.modulabs_project.data
)

SELECT
  u.CustomerID,
  DATE_DIFF(m.most_recent_date, u.InvoiceDay, DAY) AS recency
FROM user_last_purchase u
CROSS JOIN most_recent m
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON

행	CustomerID	recency
1	12662	0
2	15344	0
3	17364	0
4	15694	0
5	16626	0
6	16954	0
7	18102	0
8	12423	0
9	14422	0
10	12713	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM symbolic-folio-470301-r6.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON

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

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM symbolic-folio-470301-r6.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 시각화 | JSON

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

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

```
CREATE OR REPLACE TABLE symbolic-folio-470301-r6.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM symbolic-folio-470301-r6.modulabs_project.data
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT CustomerID, SUM(Quantity) AS item_cnt
  FROM symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

← 쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13436	1	76	1
3	15520	1	314	1
4	13298	1	96	1
5	14569	1	79	1
6	15471	1	256	2
7	15195	1	1404	2
8	14204	1	72	2
9	15318	1	642	3

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
FROM symbolic-folio-470301-r6.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보 | 결과 | 시각화 | JSON

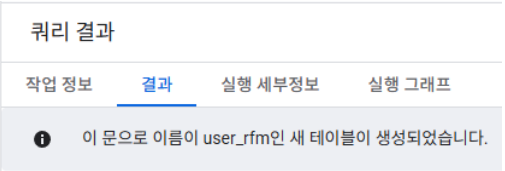
행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7
12	12358	928.1
13	12359	6183.0
14	12360	2302.1

- **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 symbolic-folio-470301-r6.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 symbolic-folio-470301-r6.modulabs_project.user_rf rf
```

```
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
  FROM symbolic-folio-470301-r6.modulabs_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 symbolic-folio-470301-r6.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	794.5
2	15520	1	314	1	343.5	343.5
3	13436	1	76	1	196.9	196.9
4	13298	1	96	1	360.0	360.0
5	14569	1	79	1	227.4	227.4
6	15471	1	256	2	454.5	454.5
7	14204	1	72	2	150.6	150.6
8	15195	1	1404	2	3861.0	3861.0
9	12650	1	250	3	242.4	242.4
10	14578	1	240	3	168.6	168.6
11	16569	1	93	3	124.2	124.2

11-8. 추가 Feature 추출

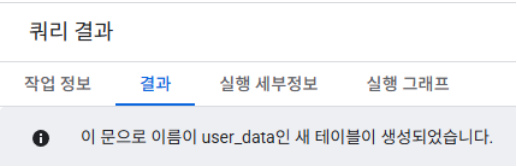
1. 구매하는 제품의 다양성

- **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 symbolic-folio-470301-r6.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM symbolic-folio-470301-r6.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM symbolic-folio-470301-r6.modulabs_project.user_rfm AS ur
```

```
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID
```

[결과 이미지를 넣어주세요]



쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	13188	1	24	11	99.6	99.6	1
2	16093	1	20	106	17.0	17.0	1
3	14351	1	12	164	51.0	51.0	1
4	13120	1	12	238	30.6	30.6	1
5	13017	1	48	7	204.0	204.0	1
6	12603	1	56	21	613.2	613.2	1
7	17102	1	2	261	25.5	25.5	1
8	18133	1	1350	212	931.5	931.5	1
9	17763	1	12	263	15.0	15.0	1
10	15940	1	4	311	35.8	35.8	1

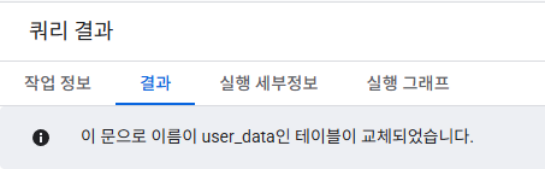
2. 평균 구매 주기

- **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 symbolic-folio-470301-r6.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
      symbolic-folio-470301-r6.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM symbolic-folio-470301-r6.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID
```

[결과 이미지를 넣어주세요]



쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	17956	1	1	249	12.8	12.8	1	0.0
2	17986	1	10	56	20.8	20.8	1	0.0
3	16990	1	100	218	179.0	179.0	1	0.0
4	16579	1	-12	365	-30.6	-30.6	1	0.0
5	13302	1	5	155	63.8	63.8	1	0.0
6	14090	1	72	324	76.3	76.3	1	0.0
7	15389	1	400	172	500.0	500.0	1	0.0
8	17443	1	504	219	534.2	534.2	1	0.0
9	18068	1	6	289	101.7	101.7	1	0.0
10	13135	1	4300	196	3096.0	3096.0	1	0.0
11	16061	1	-1	269	-29.9	-29.9	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 symbolic-folio-470301-r6.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT IF(STARTS_WITH(InvoiceNo, 'C'), InvoiceNo, NULL)) AS cancel_frequency
  FROM symbolic-folio-470301-r6.modulabs_project.data
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  ROUND(IFNULL(t.cancel_frequency, 0) / NULLIF(t.total_transactions, 0), 2) AS cancel_rate
FROM symbolic-folio-470301-r6.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

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 symbolic-folio-470301-r6.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13366	1	144	50	56.2	56.2	1	0.0	1	0	0.0
2	15657	1	24	22	30.0	30.0	1	0.0	1	0	0.0
3	12603	1	56	21	613.2	613.2	1	0.0	1	0	0.0
4	12917	1	120	128	594.0	594.0	2	0.0	1	0	0.0
5	13228	1	200	81	358.0	358.0	2	0.0	1	0	0.0
6	13514	1	40	73	152.2	152.2	4	0.0	1	0	0.0
7	17709	1	17	169	137.5	137.5	7	0.0	1	0	0.0
8	14727	1	62	274	268.6	268.6	10	0.0	1	0	0.0
9	12738	1	147	372	137.4	137.4	10	0.0	1	0	0.0
10	13806	1	56	63	256.4	256.4	10	0.0	1	0	0.0
11	15523	1	132	84	412.0	412.0	10	0.0	1	0	0.0
12	17824	1	408	51	298.4	298.4	13	0.0	1	0	0.0
13	16756	1	106	214	239.4	239.4	13	0.0	1	0	0.0
14	12581	1	160	39	179.7	179.7	14	0.0	1	0	0.0
15	16159	1	236	281	348.2	348.2	14	0.0	1	0	0.0
16	15319	1	175	283	204.5	204.5	16	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep : 테이블 구조를 유지하면서 원하는 결과를 정확히 도출한 점

Problem : 중복 컬럼 생성 및 테이블 참조 순서로 인한 오류 발생

Try : 서브쿼리와 컬럼 명시를 명확히 하여 오류 없이 구조를 유지하는 방법 시도