고객을 세그먼테이션하자 [프로젝트] (손호진)

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM euphoric-grin-466601-j7.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(*) AS `Total rows`
FROM euphoric-grin-466601-j7.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	Total rows
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 euphoric-grin-466601-j7.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	COUNT_UnitPrice	COUNT_Custome...	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 I,
```

```
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data
UNION ALL
SELECT
    'StockCode' AS S,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data
UNION ALL
SELECT
    'Description' AS D,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data
UNION ALL
SELECT
    'Quantity' AS Q,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data
UNION ALL
SELECT
    'InvoiceDate' AS ID,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data
UNION ALL
SELECT
    'UnitPrice' AS U,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data
UNION ALL
SELECT
    'CustomerID' AS CI,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data
UNION ALL
SELECT
    'Country' AS C,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM euphoric-grin-466601-j7.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

결측치 처리 전략

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

```
SELECT Description
FROM euphoric-grin-466601-j7.modulabs_project.data
WHERE StockCode = '85123A'
GROUP BY 1
ORDER BY 1;
```

[결과 이미지를 넣어주세요]

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

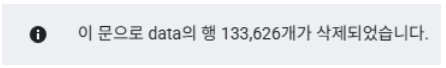
결측치 처리

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

```
DELETE FROM euphoric-grin-466601-j7.modulabs_project.data
WHERE Description IS NULL
  OR CustomerID IS NULL;
```

[결과 이미지를 넣어주세요]

- Description IS NULL 을 먼저 따로 시행한 뒤 CustomerID를 추가함에 따라 삭제 행 개수가 다르게 보임.



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 `중복` AS(
  SELECT Country, CustomerID, InvoiceDate, UnitPrice, Quantity, StockCode, InvoiceNo, Description,
    COUNT(*) AS `중복_카운트`
  FROM euphoric-grin-466601-j7.modulabs_project.data
  GROUP BY Country, CustomerID, InvoiceDate, UnitPrice, Quantity, StockCode, InvoiceNo, Description
  HAVING COUNT(*) > 1
)
SELECT COUNT(*) AS `중복된 행의 수`
FROM `중복`;
```

[결과 이미지를 넣어주세요]

행	중복된 행의 수
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 `euphoric-grin-466601-j7.modulabs_project.data` AS (
  SELECT DISTINCT *
  FROM `euphoric-grin-466601-j7.modulabs_project.data`
)
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다.

행	f0_
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS `고유 InvoiceNo 개수`
FROM `euphoric-grin-466601-j7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	고유 InvoiceNo 개수
1	22190

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo AS `고유 InvoiceNo`
FROM `euphoric-grin-466601-j7.modulabs_project.data`
ORDER BY InvoiceNo
LIMIT 100;
```

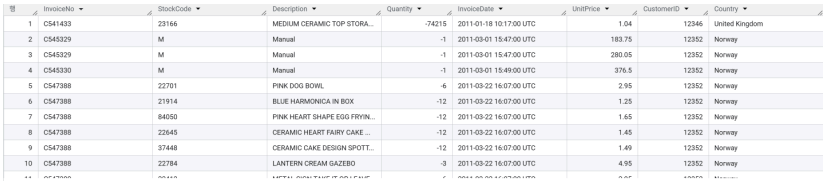
[결과 이미지를 넣어주세요]

행	고유 InvoiceNo
1	536365
2	536366
3	536367
4	536368
5	536369
6	536370
7	536371
8	536372
9	536373
10	536374
11	536375

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

```
SELECT *,
FROM `euphoric-grin-466601-j7.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

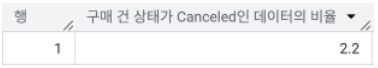


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
6	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
7	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
8	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
9	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
10	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway

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

```
SELECT ROUND(SUM(CASE WHEN # [[YOUR QUERY]] THEN 1 ELSE 0 END)/ # [[YOUR QUERY]], 1)
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



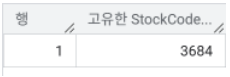
행	구매 건 상태가 Canceled인 데이터의 비율
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS `고유한 StockCode 개수`
FROM `euphoric-grin-466601-j7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

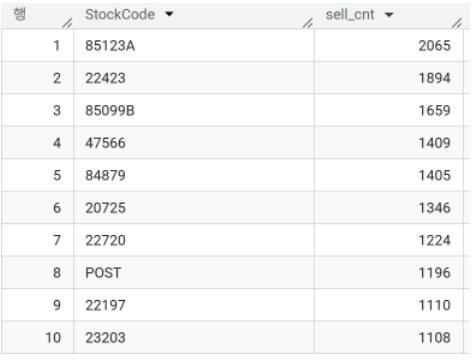


행	고유한 StockCode...
1	3684

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1
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 `euphoric-grin-466601-j7.modulabs_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 ROUND(SUM(CASE
        WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1)
        THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
        AS `숫자가 0-1개인 데이터의 비율`
FROM `euphoric-grin-466601-j7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	숫자가 0-1개인 데이터의 비율
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `euphoric-grin-466601-j7.modulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `euphoric-grin-466601-j7.modulabs_project.data`
  )
  WHERE number_count IN (0, 1)
);
```

[결과 이미지를 넣어주세요]

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 `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1
ORDER BY 1 DESC
LIMIT 30;
```

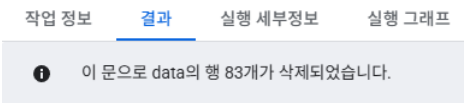
[결과 이미지를 넣어주세요]

행	Description	description_cnt
1	ZINC WIRE SWEETHEART LETTER TRAY	20
2	ZINC WIRE KITCHEN ORGANISER	12
3	ZINC WILLIE WINKIE CANDLE STICK	192
4	ZINC TOP 2 DOOR WOODEN SHELF	11
5	ZINC T-LIGHT HOLDER STARS SMALL	241
6	ZINC T-LIGHT HOLDER STARS LARGE	2
7	ZINC T-LIGHT HOLDER STAR LARGE	151
8	ZINC SWEETHEART WIRE LETTER RACK	79
9	ZINC SWEETHEART SOAP DISH	19
10	ZINC STAR T-LIGHT HOLDER	1
11	ZINC PLANT POT HOLDER	1
12	ZINC METAL HEART DECORATION	488
13	ZINC HERB GARDEN CONTAINER	192
14	ZINC HEARTS PLANT POT HOLDER	50
15	ZINC HEART LATTICE TRAY OVAL	7
16	ZINC HEART LATTICE T-LIGHT HOLDER	63
17	ZINC HEART LATTICE CHARGER SMALL	8
18	ZINC HEART LATTICE CHARGER LARGE	14
19	ZINC HEART LATTICE 2 WALL PLANTER	11
20	ZINC HEART FLOWER T-LIGHT HOLDER	57
21	ZINC FOLKART SLEIGH BELLS	350

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `euphoric-grin-466601-j7.modulabs_project.data`
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```

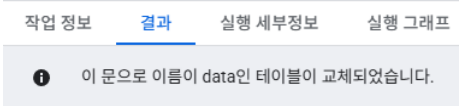
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `euphoric-grin-466601-j7.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `euphoric-grin-466601-j7.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 `euphoric-grin-466601-j7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	min_price	max_price	avg_price
1	0.0	649.5	2.9049567574059725

- 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 `euphoric-grin-466601-j7.modulabs_project.data`
WHERE UnitPrice = 0;
```

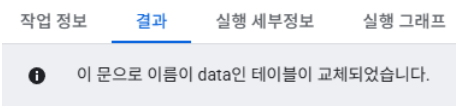
[결과 이미지를 넣어주세요]

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 `euphoric-grin-466601-j7.modulabs_project.data` AS
SELECT *
FROM `euphoric-grin-466601-j7.modulabs_project.data`
WHERE UnitPrice != 0;
```

[결과 이미지를 넣어주세요]



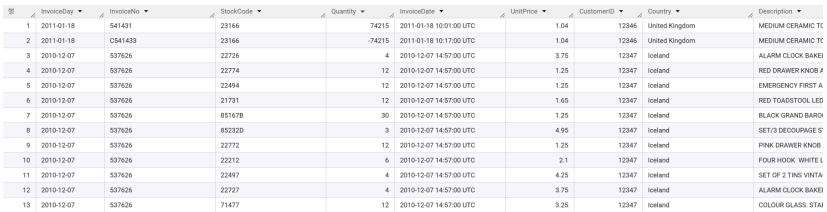
11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `euphoric-grin-466601-j7.modulabs_project.data`
```

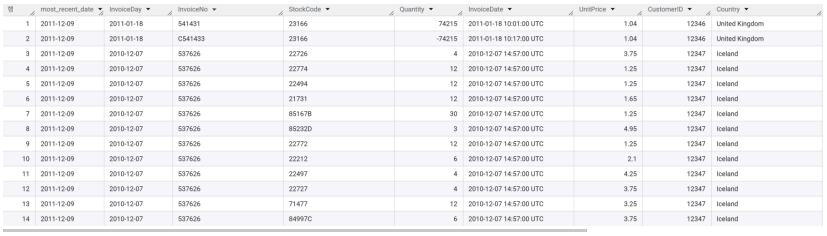
[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT (
  SELECT MAX(DATE(InvoiceDate))
  FROM `euphoric-grin-466601-j7.modulabs_project.data`
  ) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `euphoric-grin-466601-j7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-12-09	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom
2	2011-12-09	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
3	2011-12-09	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
4	2011-12-09	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
5	2011-12-09	2010-12-07	537626	22494	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
6	2011-12-09	2010-12-07	537626	21731	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland
7	2011-12-09	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
8	2011-12-09	2010-12-07	537626	85232D	3	2010-12-07 14:57:00 UTC	4.95	12347	Iceland
9	2011-12-09	2010-12-07	537626	22772	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
10	2011-12-09	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC	2.1	12347	Iceland
11	2011-12-09	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland
12	2011-12-09	2010-12-07	537626	22727	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
13	2011-12-09	2010-12-07	537626	71477	12	2010-12-07 14:57:00 UTC	3.25	12347	Iceland
14	2011-12-09	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland

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

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

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
17	12363	2011-08-22
18	12364	2011-12-02
19	12365	2011-02-21
20	12367	2011-12-05
21	12370	2011-10-19

- **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]

행	CustomerID	recency
1	12697	21
2	12721	31
3	13113	0
4	13398	192
5	13430	18
6	13471	1
7	13577	25
8	13662	85
9	13856	168
10	14377	191
11	14562	3
12	14577	12
13	14665	60
14	14796	1
15	15384	169
16	15482	15
17	15570	7
18	15988	19
19	16037	269
20	16038	7
21	16244	211

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

```
CREATE OR REPLACE TABLE `euphoric-grin-466601-j7.modulabs_project.user_r` AS
SELECT CustomerID,
      EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
    FROM `euphoric-grin-466601-j7.modulabs_project.data`
    GROUP BY 1
    )
ORDER BY recency;
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	12662	0
2	17364	0
3	16954	0
4	13777	0
5	12748	0
6	14397	0
7	14441	0
8	13113	0
9	13069	0
10	15910	0
11	17315	0
12	12526	0
13	17754	0
14	12518	0
15	14446	0
16	14051	0
17	13426	0
18	18102	0
19	17428	0
20	17001	0
21	17389	0
22	12985	0
23	15804	0
24	12680	0
25	17581	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

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
15	12361	9
16	12362	264
17	12363	23
18	12364	81
19	12365	21
20	12367	10
21	12370	165

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

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
15	12361	90
16	12362	2180
17	12363	408
18	12364	1499
19	12365	173
20	12367	172
21	12370	2349

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

```
CREATE OR REPLACE TABLE euphoric-grin-466601-j7.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1
)

-- 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 euphoric-grin-466601-j7.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID
ORDER BY purchase_cnt;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency
1	12603	1	56	21
2	12791	1	96	373
3	12814	1	48	101
4	12943	1	-1	301
5	13017	1	48	7
6	13099	1	288	99
7	13120	1	12	238
8	13135	1	4300	196
9	13185	1	12	267
10	13188	1	24	11
11	13270	1	200	366
12	13302	1	5	155
13	13307	1	4	120
14	13366	1	144	50
15	13391	1	4	203
16	13703	1	10	318
17	13747	1	8	373
18	13829	1	-12	359
19	13841	1	100	252
20	14090	1	72	324
21	14119	1	-2	354
22	14351	1	12	164
23	14424	1	48	17
24	14576	1	12	372
25	14679	1	-1	371

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1;
```

[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
11	12357	6208.0
12	12358	928.0
13	12359	6183.0
14	12360	2302.0
15	12361	175.0
16	12362	4666.0
17	12363	552.0
18	12364	1208.0
19	12365	321.0
20	12367	151.0
21	12370	3422.0

- **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 euphoric-grin-466601-j7.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), 0) AS user_average
FROM euphoric-grin-466601-j7.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
FROM `euphoric-grin-466601-j7.modulabs_project.data`
GROUP BY 1
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	16446	4	2	0	3.0	1.0
2	14446	276	856	0	1006.0	4.0
3	15910	266	1013	0	1229.0	5.0
4	12748	4440	23516	0	29820.0	7.0
5	13069	469	5454	0	3713.0	8.0
6	17364	409	2671	0	4437.0	11.0
7	17315	482	3805	0	6153.0	13.0
8	12423	118	1312	0	1624.0	14.0
9	15804	273	2513	0	3849.0	14.0
10	15344	32	106	0	477.0	15.0
11	12518	119	1306	0	1841.0	15.0
12	12985	78	1413	0	1216.0	16.0
13	12662	222	2023	0	3511.0	16.0
14	12680	49	439	0	791.0	16.0
15	12526	68	624	0	1173.0	17.0
16	16558	474	5346	0	8257.0	17.0
17	17754	90	1767	0	1632.0	18.0
18	14422	222	2906	0	4264.0	19.0
19	12713	37	505	0	795.0	21.0
20	17490	85	1022	0	1937.0	23.0
21	13426	158	2225	0	3558.0	23.0
22	16626	184	2670	0	4380.0	24.0
23	17001	169	2164	0	3990.0	24.0
24	15311	2478	37673	0	59284.0	24.0
25	17581	451	5849	0	10716.0	24.0

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM `euphoric-grin-466601-j7.modulabs_project.user_rfm`
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	16446	4	2	0	3.0	1.0
2	14446	276	856	0	1006.0	4.0
3	15910	266	1013	0	1229.0	5.0
4	12748	4440	23516	0	29820.0	7.0
5	13069	469	5454	0	3713.0	8.0
6	17364	409	2671	0	4437.0	11.0
7	17315	482	3805	0	6153.0	13.0
8	12423	118	1312	0	1624.0	14.0
9	15804	273	2513	0	3849.0	14.0
10	15344	32	106	0	477.0	15.0
11	12518	119	1306	0	1841.0	15.0
12	12985	78	1413	0	1216.0	16.0
13	12662	222	2023	0	3511.0	16.0
14	12680	49	439	0	791.0	16.0
15	12526	68	624	0	1173.0	17.0
16	16558	474	5346	0	8257.0	17.0
17	17754	90	1767	0	1632.0	18.0
18	14422	222	2906	0	4264.0	19.0
19	12713	37	505	0	795.0	21.0
20	17490	85	1022	0	1937.0	23.0
21	13426	158	2225	0	3558.0	23.0
22	16626	184	2670	0	4380.0	24.0
23	17001	169	2164	0	3990.0	24.0
24	15311	2478	37673	0	59284.0	24.0
25	17581	451	5849	0	10716.0	24.0

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;
```

[결과 이미지를 넣어주세요]

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15389	1	400	172	500.0	500.0	1
2	12346	2	0	325	0.0	0.0	1
3	13068	2	200	10	344.0	172.0	1
4	16428	1	-1	81	-3.0	-3.0	1
5	16144	1	16	246	175.0	175.0	1
6	17925	1	72	372	244.0	244.0	1
7	18233	1	4	325	440.0	440.0	1
8	18174	1	50	7	104.0	104.0	1
9	16990	1	100	218	179.0	179.0	1
10	13188	1	24	11	100.0	100.0	1
11	16078	1	16	283	79.0	79.0	1
12	16998	5	74	149	288.0	58.0	1
13	13703	1	10	318	100.0	100.0	1
14	18017	3	300	81	523.0	174.0	1
15	17616	3	224	173	571.0	190.0	1
16	14119	1	-2	354	-20.0	-20.0	1
17	16881	1	600	66	432.0	432.0	1
18	18268	2	0	134	0.0	0.0	1
19	15753	1	144	304	79.0	79.0	1
20	18184	1	60	15	50.0	50.0	1
21	13366	1	144	50	56.0	56.0	1
22	15118	1	1440	134	245.0	245.0	1
23	15510	1	2	330	250.0	250.0	1
24	15070	1	36	372	106.0	106.0	1
25	17331	1	16	123	175.0	175.0	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 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;
```

[결과 이미지를 넣어주세요]

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_interval
1	17925	1	72	372	244.0	244.0	1	0.0
2	14576	1	12	372	35.0	35.0	1	0.0
3	15524	1	4	24	440.0	440.0	1	0.0
4	17752	1	192	359	81.0	81.0	1	0.0
5	15488	1	72	92	76.0	76.0	1	0.0
6	15389	1	400	172	500.0	500.0	1	0.0
7	13099	1	288	99	207.0	207.0	1	0.0
8	16737	1	288	53	418.0	418.0	1	0.0
9	16323	1	50	196	208.0	208.0	1	0.0
10	16093	1	20	106	17.0	17.0	1	0.0
11	13135	1	4300	196	3096.0	3096.0	1	0.0
12	13307	1	4	120	15.0	15.0	1	0.0
13	17307	1	-144	365	-153.0	-153.0	1	0.0
14	15668	1	72	217	76.0	76.0	1	0.0
15	17382	1	24	65	50.0	50.0	1	0.0
16	13829	1	-12	359	-102.0	-102.0	1	0.0
17	14119	1	-2	354	-20.0	-20.0	1	0.0
18	12603	1	56	21	613.0	613.0	1	0.0
19	15562	1	39	351	135.0	135.0	1	0.0
20	12943	1	-1	301	-4.0	-4.0	1	0.0
21	16738	1	3	297	4.0	4.0	1	0.0
22	15657	1	24	22	30.0	30.0	1	0.0
23	15940	1	4	311	36.0	36.0	1	0.0
24	18233	1	4	325	440.0	440.0	1	0.0
25	17102	1	2	261	26.0	26.0	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 euphoric-grin-466601-j7.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS cancel_frequency
  FROM euphoric-grin-466601-j7.modulabs_project.data
  GROUP BY 1
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND((t.cancel_frequency / t.total_transactions), 2) AS cancel_rate
FROM `euphoric-grin-466601-j7.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13307	1	4	120	15.0	15.0	1	0.0	1	0	0.0
2	14705	1	100	198	179.0	179.0	1	0.0	1	0	0.0
3	17715	1	384	200	326.0	326.0	1	0.0	1	0	0.0
4	16856	2	12	14	35.0	18.0	2	0.0	1	0	0.0
5	12548	4	48	166	67.0	17.0	4	0.0	1	0	0.0
6	16959	7	49	87	117.0	17.0	7	0.0	1	0	0.0
7	15958	7	33	74	104.0	15.0	7	0.0	1	0	0.0
8	17887	10	30	129	111.0	11.0	10	0.0	1	0	0.0
9	15388	10	88	270	141.0	14.0	10	0.0	1	0	0.0
10	16337	10	87	36	151.0	15.0	10	0.0	1	0	0.0
11	14241	10	146	183	214.0	21.0	10	0.0	1	0	0.0
12	13508	12	24	243	111.0	9.0	12	0.0	1	0	0.0
13	15607	13	202	336	105.0	8.0	13	0.0	1	0	0.0
14	13065	14	74	373	206.0	15.0	14	0.0	1	0	0.0
15	17153	14	104	24	214.0	15.0	14	0.0	1	0	0.0
16	12519	16	172	63	277.0	17.0	16	0.0	1	0	0.0
17	13781	17	30	250	128.0	8.0	17	0.0	1	0	0.0
18	15341	17	812	80	2021.0	119.0	17	0.0	1	0	0.0
19	12648	20	159	154	300.0	15.0	20	0.0	1	0	0.0
20	15243	22	185	74	317.0	14.0	22	0.0	1	0	0.0
21	13485	23	277	233	454.0	20.0	23	0.0	1	0	0.0
22	17274	25	77	35	108.0	4.0	24	0.0	1	0	0.0
23	16272	25	224	116	412.0	16.0	25	0.0	1	0	0.0
24	13398	26	281	192	457.0	18.0	26	0.0	1	0	0.0

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

```
SELECT *
FROM euphoric-grin-466601-j7.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13307	1	4	120	15.0	15.0	1	0.0	1	0	0.0
2	14705	1	100	198	179.0	179.0	1	0.0	1	0	0.0
3	17715	1	384	200	326.0	326.0	1	0.0	1	0	0.0
4	16856	2	12	14	35.0	18.0	2	0.0	1	0	0.0
5	12548	4	48	166	67.0	17.0	4	0.0	1	0	0.0
6	16959	7	49	87	117.0	17.0	7	0.0	1	0	0.0
7	15958	7	33	74	104.0	15.0	7	0.0	1	0	0.0
8	17887	10	30	129	111.0	11.0	10	0.0	1	0	0.0
9	15388	10	88	270	141.0	14.0	10	0.0	1	0	0.0
10	16337	10	87	36	151.0	15.0	10	0.0	1	0	0.0
11	14241	10	146	183	214.0	21.0	10	0.0	1	0	0.0
12	13508	12	24	243	111.0	9.0	12	0.0	1	0	0.0
13	15607	13	202	336	105.0	8.0	13	0.0	1	0	0.0
14	13065	14	74	373	206.0	15.0	14	0.0	1	0	0.0
15	17153	14	104	24	214.0	15.0	14	0.0	1	0	0.0
16	12519	16	172	63	277.0	17.0	16	0.0	1	0	0.0
17	13781	17	30	250	128.0	8.0	17	0.0	1	0	0.0
18	15341	17	812	80	2021.0	119.0	17	0.0	1	0	0.0
19	12648	20	159	154	300.0	15.0	20	0.0	1	0	0.0
20	15243	22	185	74	317.0	14.0	22	0.0	1	0	0.0
21	13485	23	277	233	454.0	20.0	23	0.0	1	0	0.0
22	17274	25	77	35	108.0	4.0	24	0.0	1	0	0.0
23	16272	25	224	116	412.0	16.0	25	0.0	1	0	0.0
24	13398	26	281	192	457.0	18.0	26	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep : 서브쿼리, WITH 문에 익숙해짐.

Problem : 데이터 삭제 시 되돌리기 어려워, 새롭게 data 파일을 써야하는 경우 발생.

Try : sql 파일로 저장하여 git 으로 관리하면 자료 수정이 용이해보임.