고객을 세그먼테이션하자 [프로젝트] 최효근

11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기



[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT COUNT(*) AS total_rows
FROM `artful-striker-473603-f9.modulabs_project.data`;

[결과 이미지를 넣어주세요]



데이터 수 세기

• COUNT 함수를 사용해서, 각 컬럼별 데이터 포인트의 수를 세어 보기

SELECT
COUNT(InvoiceNo) AS InvoiceNo_count,
COUNT(StockCode) AS StockCode_count,
COUNT(Description) AS Description_count,
COUNT(Quantity) AS Quantity_count,
COUNT(InvoiceDate) AS InvoiceDate_count,
COUNT(UnitPrice) AS UnitPrice_count,
COUNT(CustomerID) AS CustomerID_count,
COUNT(Country) AS Country_count
FROM `artful-striker-473603-f9.modulabs_project.data`;

[결과 이미지를 넣어주세요]



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 InvoiceNo, COUNT(*) - COUNT(InvoiceNo) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`

UNION ALL

SELECT 'StockCode' AS StockCode, COUNT(*) - COUNT(StockCode) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`

UNION ALL

SELECT 'Description' AS Description, COUNT(*) - COUNT(Description) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`

UNION ALL

SELECT 'Quantity' AS Quantity, COUNT(*) - COUNT(Quantity) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`

UNION ALL

SELECT 'InvoiceDate' AS InvoiceDate, COUNT(*) - COUNT(InvoiceDate) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`

UNION ALL

SELECT 'UnitPrice' AS UnitPrice, COUNT(*) - COUNT(UnitPrice) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`

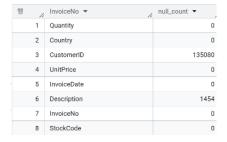
UNION ALL

SELECT 'CustomerID' AS CustomerID, COUNT(*) - COUNT(CustomerID) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`

UNION ALL

SELECT 'Country' AS Country, COUNT(*) - COUNT(Country) AS null_count FROM `artful-striker-473603-f9.modulabs_project.data`;

[결과 이미지를 넣어주세요]



결측치 처리 전략

• StockCode = '85123A' 의 Description 을 추출하는 쿼리문을 작성하기

SELECT Description FROM `artful-striker-473603-f9.modulabs_project.data` WHERE StockCode = '85123A';



결측치 처리

• DELETE 구문을 사용하며, WHERE 절을 통해 데이터를 제거할 조건을 제시

DELETE FROM `artful-striker-473603-f9.modulabs_project.data` WHERE StockCode = '85123A';

[결과 이미지를 넣어주세요]

● 이 문으로 data의 행 2,313개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

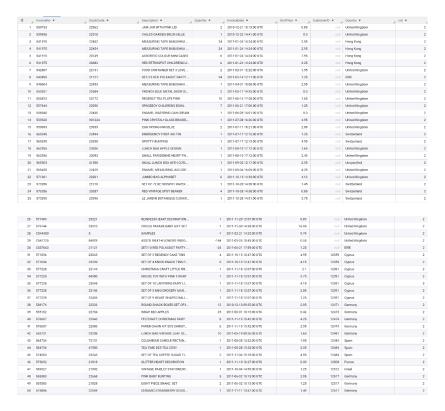
- 중복된 행의 수를 세어보기
 - 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

SELECT *
FROM project_name.modulabs_project.data
[[YOUR QUERY]]

SELECT
InvoiceNo,
StockCode,
Description,
Quantity,
InvoiceDate,
UnitPrice,
CustomerID,
Country,

```
COUNT(*) AS cnt
FROM `artful-striker-473603-f9.modulabs_project.data`
GROUP BY
InvoiceNo,
StockCode,
Description,
Quantity,
InvoiceDate,
UnitPrice,
CustomerID,
Country
HAVING COUNT(*) > 1;
```

[결과 이미지를 넣어주세요]



중복값 처리

• 중복값을 제거하는 쿼리문 작성하기

[결과 이미지를 넣어주세요]

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

```
SELECT DISTINCT *
FROM `artful-striker-473603-f9.modulabs_project.data`
LIMIT 100;

CREATE OR REPLACE TABLE `artful-striker-473603-f9.modulabs_project.data` AS
SELECT DISTINCT *
FROM `artful-striker-473603-f9.modulabs_project.data`;
```

고객을 세그먼테이션하자 [프로젝트] 최효근



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 'artful-striker-473603-f9.modulabs_project.data';

[결과 이미지를 넣어주세요]



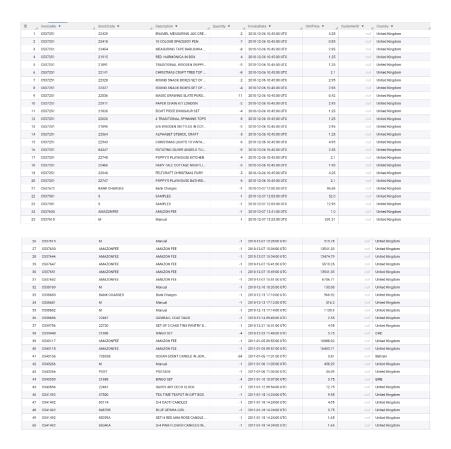
• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo FROM `artful-striker-473603-f9.modulabs_project.data` LIMIT 100;



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

SELECT *
FROM `artful-striker-473603-f9.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;



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

SELECT
ROUND(
100 * SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*),
1
) AS canceled_ratio_percent
FROM `artful-striker-473603-f9.modulabs_project.data`;

[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count FROM `artful-striker-473603-f9.modulabs_project.data`;

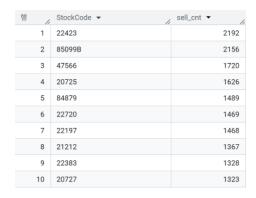
[결과 이미지를 넣어주세요]



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

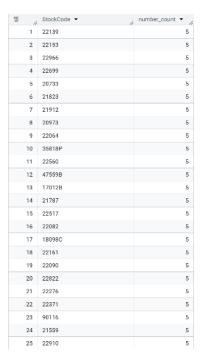
SELECT StockCode, COUNT(*) AS sell_cnt FROM `artful-striker-473603-f9.modulabs_project.data` GROUP BY StockCode ORDER BY sell_cnt DESC LIMIT 10;

[결과 이미지를 넣어주세요]



- 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 `artful-striker-473603-f9.modulabs_project.data`
)
WHERE number_count >= LENGTH(StockCode) - 1; -- 숫자가 0~1개인 경우
```





- 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(
100 * SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1
THEN 1 ELSE 0 END)
/ COUNT(*), 2
) AS zero_one_number_ratio_percent
FROM `artful-striker-473603-f9.modulabs_project.data`;
```

행 zero_one_number... / 1 0.55

• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `artful-striker-473603-f9.modulabs_project.data`
WHERE StockCode IN (
SELECT DISTINCT StockCode
FROM (
SELECT StockCode
FROM `artful-striker-473603-f9.modulabs_project.data`
WHERE LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) = LENGTH(StockCode) -- 숫자가 전혀 없는 코드
)
);
```

[결과 이미지를 넣어주세요]

● 이 문으로 data의 행 2,790개가 삭제되었습니다.

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT Description, COUNT(*) AS description_cnt FROM `artful-striker-473603-f9.modulabs_project.data` GROUP BY Description ORDER BY description_cnt DESC LIMIT 30;

[결과 이미지를 넣어주세요]



• 서비스 관련 정보를 포함하는 행들을 제거하기

SELECT DISTINCT Description
FROM 'artful-striker-473603-f9.modulabs_project.data'
WHERE UPPER(Description) LIKE '%TEST%'
OR UPPER(Description) LIKE '%ADJUST%'
OR UPPER(Description) LIKE '%SAMPLE%'
OR UPPER(Description) LIKE '%POSTAGE%';



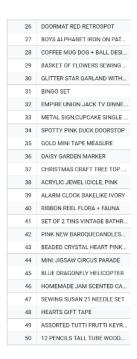
• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `artful-striker-473603-f9.modulabs_project.data` AS
SELECT

* EXCEPT (Description),
UPPER(Description) AS Description
FROM `artful-striker-473603-f9.modulabs_project.data`;

SELECT Description
FROM `artful-striker-473603-f9.modulabs_project.data`
LIMIT 100;
```





UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

SELECT
MIN(UnitPrice) AS min_price,
MAX(UnitPrice) AS max_price,
ROUND(AVG(UnitPrice), 2) AS avg_price
FROM `artful-striker-473603-f9.modulabs_project.data`;

[결과 이미지를 넣어주세요]



• 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,
ROUND(AVG(Quantity), 2) AS avg_quantity
FROM `artful-striker-473603-f9.modulabs_project.data`
WHERE UnitPrice = 0;

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE `artful-striker-473603-f9.modulabs_project.data` AS SELECT *
FROM `artful-striker-473603-f9.modulabs_project.data`
WHERE UnitPrice <> 0;

SELECT UnitPrice, Quantity, StockCode, Description FROM `artful-striker-473603-f9.modulabs_project.data` LIMIT 100;

[결과 이미지를 넣어주세요]

행 .	UnitPrice ▼	Quantity •	StockCode ▼	Description ▼
1	UnitPrice ▼ 2.98	Quantity • //	22949	36 DOILIES DOLLY GIRL
2	1.49	1	84535B	FAIRY CAKES NOTEBOOK A6 SI
3	14.43	1	20685	DOORMAT RED RETROSPOT
4	5.91	1	22470	HEART OF WICKER LARGE
5	0.85	4	22419	LIPSTICK PEN RED
6	4.21	1	22182	CAKE STAND VICTORIAN FILIG
7	3.36	4	22198	LARGE POPCORN HOLDER
8	3.36	2	22174	PHOTO CUBE
9	2.51	3	17012A	ORIGAMI VANILLA INCENSE/C
10	14.43	1 2	21523	DOORMAT FANCY FONT HOME RED RETROSPOT MUG
- 11				
12	0.85	4	22539	MINI JIGSAW DOLLY GIRL
13	0.85	1	21703	BAG 125G SWIRLY MARBLES
14	4.21	1	22876	NUMBER TILE COTTAGE GARD
15	2.51	2	22095	LADS ONLY TISSUE BOX
16	2.98	1	21210	SET OF 72 RETROSPOT PAPER
17	0.85	2	85131B	BEADED CRYSTAL HEART GREE
18	7.62	5	21479	WHITE SKULL HOT WATER BOT
19	1.28	1	21975	PACK OF 60 DINOSAUR CAKE C
20	8.47	2	85132A	CHARLIE + LOLA BISCUITS TINS
21	3.36	1	21874	GIN AND TONIC MUG
22	2.51	1	90116	FRUIT SALAD BAG CHARM
23	1.66	1	85177	BASKET OF FLOWERS SEWING
24	2.51	1	82613B	METAL SIGN, CUPCAKE SINGLE
25	10.17	2	85127	SMALL SQUARE CUT GLASS CA
26	0.85	1	22153	ANGEL DECORATION STARS O
27	1.26	1	90071	RUBY GLASS CLUSTER EARRIN
28	7.62	2	22550	HOLIDAY FUN LUDO
29	3.36	2	72799E	IVORY PILLAR CANDLE SILVER
30	8.47	3	22114	HOT WATER BOTTLE TEA AND
31	11.87	2	21662	VINTAGE GLASS COFFEE CADDY
32	0.85	1	90214D	LETTER "D" BLING KEY RING
33	5.91	1	22314	OFFICE MUG WARMER CHOC+
34	3.36	3	22077	6 RIBBONS RUSTIC CHARM
35	3.36	1	21935	SUKI SHOULDER BAG
36	5.91	4	22910	PAPER CHAIN KIT VINTAGE CH
37	1.69	8	71459	HANGING JAM JAR T-LIGHT HO
38	12.72	1	20902	VINTAGE KEEPSAKE BOX PARI
39	2.51	1	21949	SET OF 6 STRAWBERRY CHOPS
40	0.85	8	22531	MAGIC DRAWING SLATE CIRCU
41	5.06	3	22588	CARD HOLDER GINGHAM HEART
42	1.65	6	20854	BLUE PATCH PURSE PINK HEART
43	5.91	2	22798	ANTIQUE GLASS DRESSING TA
44	0.85	1	84692	BOX OF 24 COCKTAIL PARASOLS
45	4.21	1	22380	TOY TIDY SPACEBOY
46	8.47	1	21218	RED SPOTTY BISCUIT TIN
47	2.51	3	22988	SOLDIERS EGG CUP
48	12.72	40	22423	REGENCY CAKESTAND 3 TIER
49	1.66	2	22196	SMALL HEART MEASURING SP
50	1.66	1	21884	CAKES AND BOWS GIFT TAPE

11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT

DATE(InvoiceDate) AS InvoiceDay,

*

FROM `artful-striker-473603-f9.modulabs_project.data` LIMIT 100;

[결과 이미지를 넣어주세요]

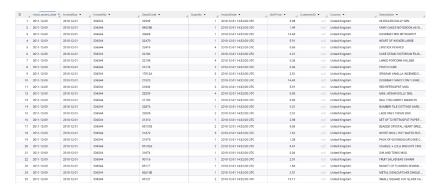


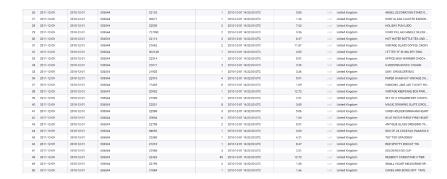
• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT

 $(SELECT\ MAX(DATE(InvoiceDate))\ FROM\ `artful-striker-473603-f9.modulabs_project.data`)\ AS\ most_recent_date, DATE(InvoiceDate)\ AS\ InvoiceDay,$

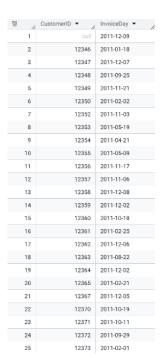
FROM `artful-striker-473603-f9.modulabs_project.data` LIMIT 100;





• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

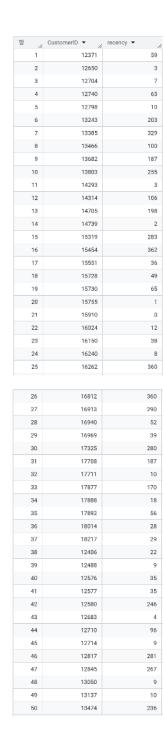
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `artful-striker-473603-f9.modulabs_project.data`
GROUP BY CustomerID;



26 12374 2011-11-14 27 12375 2011-12-07 28 12377 2011-01-28 29 12378 2011-08-02 30 12379 2011-09-19 31 12380 2011-11-18 32 12381 2011-12-05 33 12383 2011-06-08 34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-11-18 39 12393 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-04 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405			
28 12377 2011-01-28 29 12378 2011-08-02 30 12379 2011-09-19 31 12380 2011-11-18 32 12381 2011-12-05 33 12383 2011-06-08 34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-10-07 41 12394 2011-11-18 40 12394 2011-11-24 41 12395 2011-10-07 41 12395 2011-02-1 42 12397 2011-10-25 44 12399 2011-02-5 44 12399 2011-02-09 46 12402 2011-02-0 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-7	26	12374	2011-11-14
29 12378 2011-08-02 30 12379 2011-09-19 31 12380 2011-11-18 32 12381 2011-12-05 33 12383 2011-06-08 34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-11-18 39 12393 2011-09-28 40 12394 2011-11-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-02-5 44 12399 2011-02-5 44 12399 2011-02-09 46 12402 2011-01-0 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	27	12375	2011-12-07
30 12379 2011-09-19 31 12380 2011-11-18 32 12381 2011-12-05 33 12383 2011-06-08 34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-11-18 39 12393 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14	28	12377	2011-01-28
31 12380 2011-11-18 32 12381 2011-12-05 33 12383 2011-06-08 34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-01-04 37 12390 2011-09-21 38 12391 2011-11-18 39 12393 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-02-5 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	29	12378	2011-08-02
32 12381 2011-12-05 33 12383 2011-06-08 34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	30	12379	2011-09-19
38 12383 2011-06-08 34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-07-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	31	12380	2011-11-18
34 12384 2011-11-11 35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-02-5 44 12399 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	32	12381	2011-12-05
35 12386 2011-01-06 36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-11-18 39 12393 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	33	12383	2011-06-08
36 12388 2011-11-24 37 12390 2011-09-21 38 12391 2011-10-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	34	12384	2011-11-11
37 12390 2011-09-21 38 12391 2011-11-18 39 12393 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	35	12386	2011-01-06
38 12391 2011-11-18 39 12393 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	36	12388	2011-11-24
39 12393 2011-09-28 40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	37	12390	2011-09-21
40 12394 2011-10-07 41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	38	12391	2011-11-18
41 12395 2011-11-24 42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	39	12393	2011-09-28
42 12397 2011-11-04 43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	40	12394	2011-10-07
43 12398 2011-10-25 44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	41	12395	2011-11-24
44 12399 2011-08-12 45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	42	12397	2011-11-04
45 12401 2011-02-09 46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	43	12398	2011-10-25
46 12402 2011-01-20 47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	44	12399	2011-08-12
47 12403 2011-10-21 48 12405 2011-07-14 49 12406 2011-11-17	45	12401	2011-02-09
48 12405 2011-07-14 49 12406 2011-11-17	46	12402	2011-01-20
49 12406 2011-11-17	47	12403	2011-10-21
	48	12405	2011-07-14
50 12407 2011-10-21	49	12406	2011-11-17
	50	12407	2011-10-21

• 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 `artful-striker-473603-f9.modulabs_project.data`
GROUP BY CustomerID
);
```

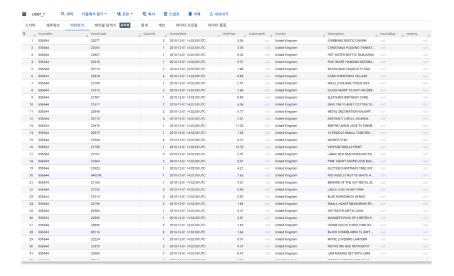


• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user_r 이라는 이름의 테이블로 저장하기

```
CREATE OR REPLACE TABLE `artful-striker-473603-f9.modulabs_project.user_r` AS
WITH cleaned_data AS (
    SELECT *
    FROM `artful-striker-473603-f9.modulabs_project.data`
    WHERE UnitPrice <> 0
),
    user_last_purchase AS (
    SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM cleaned_data
```

```
GROUP BY CustomerID
),
most_recent AS (
SELECT MAX(InvoiceDay) AS most_recent_date
 FROM user_last_purchase
),
recency_calc AS (
SELECT
 u.CustomerID,
 u.InvoiceDay,
 EXTRACT(DAY FROM m.most_recent_date - u.InvoiceDay) AS recency
 FROM user_last_purchase u
 CROSS JOIN most_recent m
SELECT
d.*,
r.InvoiceDay,
r.recency
FROM cleaned_data d
LEFT JOIN recency_calc r
ON d.CustomerID = r.CustomerID;
```

[결과 이미지를 넣어주세요]



Frequency

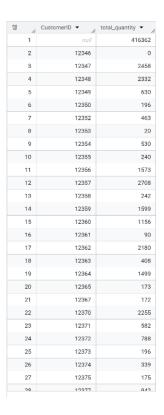
• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `artful-striker-473603-f9.modulabs_project.data`
GROUP BY CustomerID;
```

//	CustomerID	• //	purchase_cnt	• ,
1		null		1410
2		12346		2
3		12347		7
4		12348		4
5		12349		1
6 7		12350		1
8		12352 12353		8
9		12353		1
10		12355		1
11		12356		3
12		12357		1
13		12358		2
14		12359		6
15		12360		3
16		12361		1
17		12362		13
18		12363		2
19		12364		4
20		12365		1
21		12367		1
22		12370		4
23		12371		1
24		12372		3
25		12373		1
26		12374		1
27		12375		3
28		12377		2
29		12378		1
30		12379		3
31		12380		5
32		12381		4
33		12383		6
34		12384		3
35		12386		2
36		12388		6
37		12390		1
38		12391		1
39		12393		4
40		12394		2
41		12395		15
42		12397		2
43		12398		1
44		12399		4
46		12401		1
46		12402		1
47		12403		1
49		12405		3
50		12407		5
00				

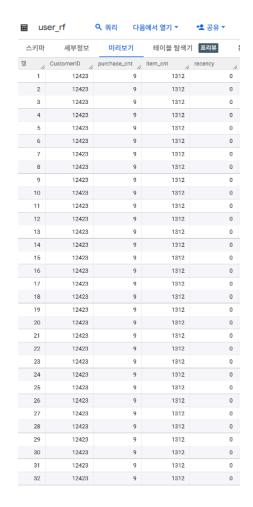
• 각 고객 별로 구매한 아이템의 총 수량 더하기

SELECT
CustomerID,
SUM(Quantity) AS total_quantity
FROM `artful-striker-473603-f9.modulabs_project.data`
GROUP BY CustomerID;



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

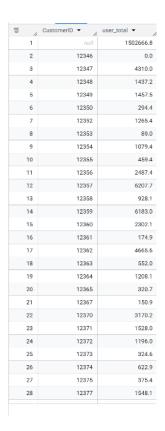
```
CREATE OR REPLACE TABLE 'artful-striker-473603-f9.modulabs_project.user_rf' AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
 FROM `artful-striker-473603-f9.modulabs_project.data`
 GROUP BY CustomerID
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
 FROM `artful-striker-473603-f9.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 `artful-striker-473603-f9.modulabs_project.user_r` AS ur
ON pc.CustomerID = ur.CustomerID;
```



Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

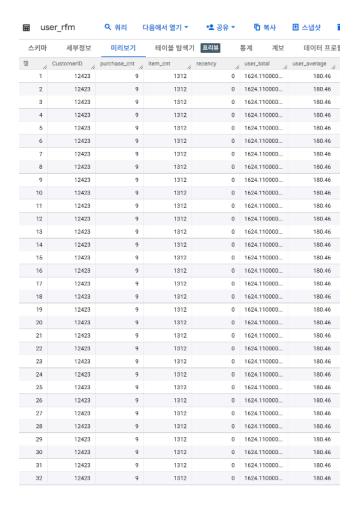
SELECT
CustomerID,
ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
FROM `artful-striker-473603-f9.modulabs_project.data`
GROUP BY CustomerID;



• 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 `artful-striker-473603-f9.modulabs_project.user_rfm` AS
 rf.CustomerID AS CustomerID,
 rf.purchase_cnt,
 rf.item_cnt,
 rf.recency,
 ut.user_total,
 ROUND(ut.user_total / rf.purchase_cnt, 2) AS user_average
FROM `artful-striker-473603-f9.modulabs_project.user_rf` rf
LEFT JOIN (
 -- 고객 별 총 지출액
 SELECT
  CustomerID,
  SUM(UnitPrice * Quantity) AS user_total
 FROM `artful-striker-473603-f9.modulabs_project.data`
 GROUP BY CustomerID
ON rf.CustomerID = ut.CustomerID;
```



RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

SELECT * FROM `artful-striker-473603-f9.modulabs_project.user_rfm` LIMIT 100;

/ C	ustomerID 🕶 🕺 purcha	se_cnt ▼ / item_	ont ▼ recen	cy ▼	user_total ▼	user_average ▼
1	12423	9	1312	0	1624.110000000	180.4
2	12423	9	1312	0	1624.110000000	180.4
3	12423	9	1312	0	1624.110000000	180.4
4	12423	9	1312	0	1624.110000000	180.4
5	12423	9	1312	0	1624.1100000000	180.4
6	12423	9	1312	0	1624.110000000	180.4
7	12423	9	1312	0	1624.110000000	180.4
8	12423	9	1312	0	1624.110000000	180.4
9	12423	9	1312	0	1624.110000000	180.4
10	12423	9	1312	0	1624.110000000	180.4
11	12423	9	1312	0	1624.110000000	180.4
12	12423	9	1312	0	1624.110000000	180.4
13	12423	9	1312	0	1624.110000000	180.4
14	12423	9	1312	0	1624.110000000	180.4
15	12423	9	1312	0	1624.110000000	180.4
16	12423	9	1312	0	1624.1100000000	180.4
17	12423	9	1312	0	1624.110000000	180.4
18	12423	9	1312	0	1624.110000000	180.4
19	12423	9	1312	0	1624.110000000	180.4
20	12423	9	1312	0	1624.110000000	180.
21	12423	9	1312	0	1624.110000000	180.
22	12423	9	1312	0	1624.110000000	180.
23	12423	9	1312	0	1624.110000000	180.
24	12423	9	1312	0	1624.110000000	180.
25	12423	9	1312	0	1624.110000000	180.
20	12420	-	1012		1024.110000000	100.
26	12423	9	1312	0	1624.1100000000	180.
27	12423	9	1312	0	1624.1100000000	180
28	12423	9	1312	0	1624.110000000	180.
29	12423	9	1312	0	1624.110000000	180.
30	12423	9	1312	0	1624.110000000	180.
31	12423	9	1312	0	1624.110000000	180.
32	12423	9	1312	0	1624.110000000	180.
33	12423	9	1312	0	1624.110000000	180.
34	12423	9	1312	0	1624.110000000	180.
35	12423	9	1312	0	1624.110000000	180.
36	12423	9	1312	0	1624.110000000	180.
37	12423	9	1312	0	1624.110000000	180.
38	12423	9	1312	0	1624.110000000	180.
39	12423	9	1312	0	1624.110000000	180.
40	12423	9	1312	0	1624.110000000	180.
41	12423	9	1312	0	1624.110000000	180.
42	12423	9	1312	0	1624.110000000	180.
43	12423	9	1312	0	1624.110000000	180.
44	12423	9	1312	0	1624.110000000	180.
45	12423	9	1312	0	1624.110000000	180.
46	12423	9	1312	0	1624.110000000	180.
47	12423	9	1312	0	1624.110000000	180.
48	12423	9	1312	0	1624.110000000	180.
49	12423	9	1312	0	1624.110000000	180.
50	12423	9	1312	0	1624.110000000	180.

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]

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_
  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;
```



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 `artful-striker-473603-f9.modulabs_project.user_data` AS

WITH TransactionInfo AS (

SELECT

CustomerID,

COUNT(*) AS total_transactions,

SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency

FROM `artful-striker-473603-f9.modulabs_project.data`

GROUP BY CustomerID
)

SELECT

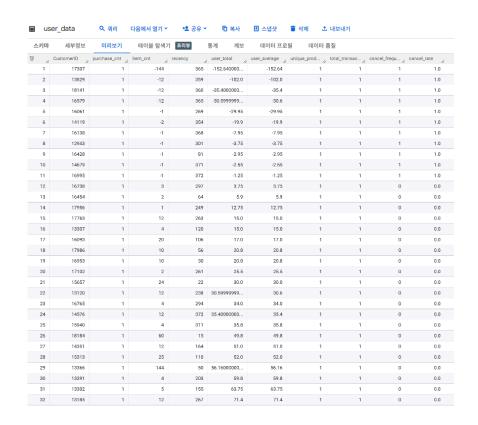
u.*, t.* EXCEPT(CustomerID),

ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate

FROM `artful-striker-473603-f9.modulabs_project.user_data` AS u

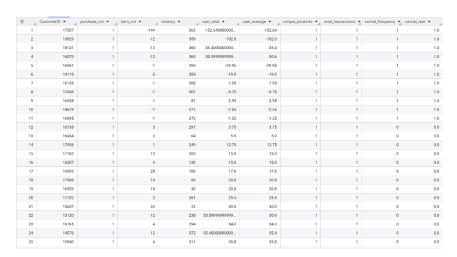
LEFT JOIN TransactionInfo AS t

ON u.CustomerID = t.CustomerID;
```



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

SELECT * FROM `artful-striker-473603-f9.modulabs_project.user_data` LIMIT 100;



26	18184	1	60	15	49.8	49.8	1	1	0	0.0
27	14351	1	12	164	51.0	51.0	1	1	0	0.0
28	15313	1	25	110	52.0	52.0	1	1	0	0.0
29	13366	1	144	50	56.160000000000	56.16	1	1	0	0.0
30	13391	1	4	203	59.8	59.8	1	1	0	0.0
31	13302	1	5	155	63.75	63.75	1	1	0	0.0
32	13185	1	12	267	71.4	71.4	1	1	0	0.0
33	18113	1	72	368	76.32000000000	76.32	1	1	0	0.0
34	15668	1	72	217	76.32000000000	76.32	1	1	0	0.0
35	16148	1	72	296	76.320000000000	76.32	1	1	0	0.0
36	14090	1	72	324	76.32000000000	76.32	1	1	0	0.0
37	15753	1	144	304	79.2	79.2	1	1	0	0.0
38	16078	1	16	283	79.2	79.2	1	1	0	0.0
39	13747	1	8	373	79.6	79.6	1	1	0	0.0
40	17752	1	192	359	80.64	80.64	1	1	0	0.0
41	13841	1	100	252	85.0	85.0	1	1	0	0.0
42	12814	1	48	101	85.92	85.92	1	1	0	0.0
43	13703	1	10	318	99.5	99.5	1	1	0	0.0
44	18068	1	6	289	101.6999999999	101.7	1	1	0	0.0
45	18174	1	50	7	104.0	104.0	1	1	0	0.0
46	15070	1	36	372	106.2	106.2	1	1	0	0.0
47	15562	1	39	351	134.55	134.55	1	1	0	0.0
48	15316	1	100	326	165.0	165.0	1	1	0	0.0
49	16144	1	16	246	175.2	175.2	1	1	0	0.0
50	17331	1	16	123	175.2	175.2	1	1	0	0.0

고객별 최근 구매일, 거래 건수, 총 지출액을 기반으로 구매 패턴을 파악할 수 있었습니다. 구매 다양성과 평균 구매 간격을 통해 충성도와 재구매 가능성을 예측할 수 있었고, 취소 빈도와 비율을 분석하여 주문 관리가 필요한 고객을 식별할 수 있었습니다.

회고

[회고 내용을 작성해주세요]

Keep:

데이터 전처리 과정에서 결측치, 중복, 단가 0원, 제품과 관련 없는 거래 기록 등 체계적으로 제거

RFM 분석 과정에서 Recency, Frequency, Monetary 지표를 개선하고, 고객별 구매 패턴을 이해하기 위해 평균 구매 간격, 구매 다양성, 취소 빈도 및 취소 비율까지 통합

MAX() 함수 관련 오류를 경험하며 서브쿼리와 집계함수 활용에 대한 이해가 깊어짐

Problem :

11-7 RFM스코어 중 '가장 최근 구매 일자를 MAX() 함수로 찾아보기' 에서 MAX() 함수와 일반 컬럼을 함께 SELECT할 때 GROUP BY 가 없어서 에러 발생 → 서브쿼리로 처리해서 오류가 안나게 해결

Try:

집계함수와 일반 컬럼을 함께 쓸 때는 서브쿼리, 윈도우 함수, GROUP BY를 적절히 활용할 것 데이터 전처리 단계에서 이상치, 결측치, 불필요한 기록을 꼼꼼히 확인하고 제거하는 습관을 유지할 것