

고객을 세그멘테이션하자 [프로젝트] 김지수

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM `ivory-partition-473602-u6.modulabs_project.data_user`  
LIMIT 10;
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536353	891234	WHITE HANGING HEART T-LIN...	6	2010-12-01 08:26:00 UTC	2.35	17850	United Kingdom
2	536355	71055	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536355	844008	CREAM CUPID HEARTS COAT R...	6	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536355	842095	KNITTED UNION FLAG HOT PA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536355	842042	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536355	227762	SET 7 BARKING BIRD FEEDER...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536355	217780	GLASS STAR FROSTED TULIPT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536355	220533	HAND BANNER UNION JACK	6	2010-12-01 08:26:00 UTC	1.85	17850	United Kingdom
9	536356	220533	HAND BANNER RED PINK DOG	6	2010-12-01 08:26:00 UTC	1.85	17850	United Kingdom
10	536357	844079	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:26:00 UTC	1.68	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT  
COUNT(*) AS total_rows  
FROM `ivory-partition-473602-u6.modulabs_project.data_user`;
```

행	total_rows
1	541909

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 `ivory-partition-473602-u6.modulabs_project.data_user`;
```

행	InvoiceNo_count	StockCode_count	Description_count	Quantity_count	InvoiceDate_count	UnitPrice_count	CustomerID_count	Country_count
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

```
-- NULL 개수 확인  
SELECT
```

```

COUNT(*) - COUNT(InvoiceNo) AS InvoiceNo_nulls,
COUNT(*) - COUNT(StockCode) AS StockCode_nulls,
COUNT(*) - COUNT(Description) AS Description_nulls,
COUNT(*) - COUNT(Quantity) AS Quantity_nulls,
COUNT(*) - COUNT(InvoiceDate) AS InvoiceDate_nulls,
COUNT(*) - COUNT(UnitPrice) AS UnitPrice_nulls,
COUNT(*) - COUNT(CustomerID) AS CustomerID_nulls,
COUNT(*) - COUNT(Country) AS Country_nulls
FROM `ivory-partition-473602-u6.modulabs_project.data_user`;

-- 컬럼별 결측치 비율
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `ivory-partition-473602-u6.modulabs_project.data_user`

UNION ALL
SELECT 'StockCode', ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `ivory-partition-473602-u6.modulabs_project.data_user`

UNION ALL
SELECT 'Description', ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `ivory-partition-473602-u6.modulabs_project.data_user`

UNION ALL
SELECT 'Quantity', ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `ivory-partition-473602-u6.modulabs_project.data_user`

UNION ALL
SELECT 'InvoiceDate', ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `ivory-partition-473602-u6.modulabs_project.data_user`

UNION ALL
SELECT 'UnitPrice', ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `ivory-partition-473602-u6.modulabs_project.data_user`

UNION ALL
SELECT 'CustomerID', ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `ivory-partition-473602-u6.modulabs_project.data_user`

UNION ALL
SELECT 'Country', ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM `ivory-partition-473602-u6.modulabs_project.data_user`;

```

행	InvoiceNo_nulls	StockCode_nulls	Description_nulls	Quantity_nulls	InvoiceDate_nulls	UnitPrice_nulls	CustomerID_nulls	Country_nulls
1	0	0	1454	0	0	0	135080	0

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM project_name.modulabs_project.data
WHERE StockCode = '85123A';
```

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

결측치 처리

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

```
DELETE FROM project_name.modulabs_project.data
WHERE CustomerID IS NULL
OR Description IS NULL;
```

i 이 문으로 data_user의 행 135,080개가 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
COUNT(*) AS duplicate_count
FROM (
  SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country,
  COUNT(*) AS cnt
  FROM project_name.modulabs_project.data
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
);
```

행	duplicate_count
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 project_name.modulabs_project.data AS
SELECT DISTINCT *
```

```
FROM project_name.modulabs_project.data;
```

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
SELECT  
COUNT(DISTINCT InvoiceNo) AS unique_invoice_count  
FROM `ivory-partition-473602-u6.modulabs_project.data_user`;
```

행	unique_invoice_c...
1	22190

- 고유한 InvoiceNo를 앞에서부터 100개를 출력하기

```
SELECT  
DISTINCT InvoiceNo  
FROM `ivory-partition-473602-u6.modulabs_project.data_user`  
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

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

```
SELECT *  
FROM project_name.modulabs_project.data  
WHERE InvoiceNo LIKE 'C%'  
LIMIT 100;
```

일	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	CS41433	21744	MEDIUM CERAMIC TOP STON...	-12	2011-01-18 18:17:00 UTC	1.56	13546	United Kingdom
2	CS41388	84030	PINK HEART SHAPED EGG FRON...	-12	2011-03-22 16:07:00 UTC	1.65	13552	Norway
3	CS41388	37468	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.69	13552	Norway
4	CS41388	22845	CERAMIC HEART FAIRY CAKE...	-12	2011-03-22 16:07:00 UTC	1.65	13552	Norway
5	CS41388	22791	PINK EGG BUN.	-4	2011-03-22 16:07:00 UTC	2.65	13552	Norway
6	CS41388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	13552	Norway
7	CS41388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	13552	Norway
8	CS41388	22813	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	13552	Norway
9	CS49155	22839	3 TIER CAKE TIN GREEN AND G...	-3	2011-04-13 13:38:00 UTC	14.65	13559	Cyprus
10	CS49155	22866	RECIPE BOOK PANTRY YELLOW...	-2	2011-04-13 13:38:00 UTC	2.95	13559	Cyprus
11	CS80165	22845	SET OF 3 REGENCY CAKE TINS	-2	2011-12-02 11:21:00 UTC	4.95	13599	Cyprus
12	CS80165	22725	SET OF 3 CAKE TINS PANTRY G...	-1	2011-12-02 11:21:00 UTC	4.95	13599	Cyprus
13	CS80165	22829	LOVE SEAL ANTIQUE WHITE T...	-1	2011-12-02 11:21:00 UTC	42.5	13599	Cyprus

페이지당 결과 수: 30 1 - 50 (전체 100개) < >

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/ COUNT(*) * 100, 1)
FROM project_name.modulabs_project.data;
```

행	fo_
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT
COUNT(DISTINCT StockCode) AS unique_stockcodes
FROM `project_name.modulabs_project.data`;
```

행	unique_stockcodes
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 project_name.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

행	StockCode	sell_cnt
1	85123A	2077
2	22423	1905
3	85099B	1662
4	84879	1418
5	47566	1416
6	20725	1359
7	22720	1232
8	POST	1196
9	20727	1126
10	22197	1118

- **StockCode**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 숫자가 0~1개인 값들에는 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 project_name.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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
SELECT
  ROUND(100 * SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1)
    THEN 1 ELSE 0 END) / COUNT(*), 2) AS percentage
FROM project_name.modulabs_project.data;
```

행	percentage
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM project_name.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM project_name.modulabs_project.data
    WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1)
  )
);
```

i 이 문으로 data_user의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM project_name.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM project_name.modulabs_project.data
WHERE
# [[YOUR QUERY]]
```

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

페이지당 결과 수: 50 ▼ 1 - 30 (전체 30행)

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT
* EXCEPT (Description),
UPPER(Description) AS Description
FROM project_name.modulabs_project.data;
```

i 이 문으로 이름이 data_user인 테이블이 교체되었습니다.

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 project_name.modulabs_project.data;
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.907457172952...

- 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 project_name.modulabs_project.data
WHERE UnitPrice = 0;
```

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

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT *
FROM project_name.modulabs_project.data
WHERE UnitPrice <> 0;
```

i 이 문으로 이름이 data_user인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541401	22146	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CREAM
2	2011-01-18	CS41403	22146	74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CREAM
3	2010-12-07	537626	849070	6	2010-12-07 14:57:00 UTC	3.75	12347	Ireland	PINK 3 PENCIL
4	2010-12-07	537626	22774	4	2010-12-07 14:57:00 UTC	3.75	12347	Ireland	ALARM CLOCK
5	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Ireland	RED DRAWER
6	2010-12-07	537626	849070	6	2010-12-07 14:57:00 UTC	3.75	12347	Ireland	BLUE 3 PENCIL
7	2010-12-07	537626	21171	12	2010-12-07 14:57:00 UTC	1.49	12347	Ireland	BATHROOM MAT
8	2010-12-07	537626	22467	4	2010-12-07 14:57:00 UTC	4.25	12347	Ireland	BOX OF 3 TRAY
9	2010-12-07	537626	84909	6	2010-12-07 14:57:00 UTC	4.25	12347	Ireland	BOX OF 6 A4B
10	2010-12-07	537626	35762	6	2010-12-07 14:57:00 UTC	5.49	12347	Ireland	CAMOUFLAGE
11	2010-12-07	537626	22466	12	2010-12-07 14:57:00 UTC	1.25	12347	Ireland	EMERGENCY

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
(SELECT MAX(InvoiceDate) FROM project_name.modulabs_project.data) AS most_recent_date,
DATE(InvoiceDate) AS InvoiceDay,
*
FROM project_name.modulabs_project.data;
```


행	project_name	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Quantity
1	2011-12-09 12:50:00 UTC	2011-01-18	581493	23166	14215	2011-01-18 10:51:00 UTC	1.04	12346	Unseal Ringtop
2	2011-12-09 12:50:00 UTC	2011-01-18	581493	23166	-74215	2011-01-18 10:51:00 UTC	1.04	12346	Unseal Ringtop
3	2011-12-09 12:50:00 UTC	2010-12-07	557626	849970	6	2010-12-07 14:51:00 UTC	3.75	12347	Kalend
4	2011-12-09 12:50:00 UTC	2010-12-07	557626	22728	4	2010-12-07 14:51:00 UTC	3.75	12347	Kalend
5	2011-12-09 12:50:00 UTC	2010-12-07	557626	22728	12	2010-12-07 14:51:00 UTC	3.75	12347	Kalend
6	2011-12-09 12:50:00 UTC	2010-12-07	557626	849970	6	2010-12-07 14:51:00 UTC	3.75	12347	Kalend
7	2011-12-09 12:50:00 UTC	2010-12-07	557626	21171	12	2010-12-07 14:51:00 UTC	1.45	12347	Kalend
8	2011-12-09 12:50:00 UTC	2010-12-07	557626	22487	4	2010-12-07 14:51:00 UTC	4.25	12347	Kalend
9	2011-12-09 12:50:00 UTC	2010-12-07	557626	84999	6	2010-12-07 14:51:00 UTC	4.25	12347	Kalend
10	2011-12-09 12:50:00 UTC	2010-12-07	557626	20782	6	2010-12-07 14:51:00 UTC	6.45	12347	Kalend
11	2011-12-09 12:50:00 UTC	2010-12-07	557626	22484	12	2010-12-07 14:51:00 UTC	1.25	12347	Kalend
12	2011-12-09 12:50:00 UTC	2010-12-07	557626	22722	12	2010-12-07 14:51:00 UTC	1.25	12347	Kalend
13	2011-12-09 12:50:00 UTC	2010-12-07	557626	22722	4	2010-12-07 14:51:00 UTC	1.75	12347	Kalend

페이지당 결과 수: 50 1 - 50 (전체 399650행) < >

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(Date(InvoiceDate)) AS InvoiceDay
FROM project_name.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

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 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
);
```

행	CustomerID	recency
1	12528	9
2	12685	28
3	12758	116
4	12863	52
5	12886	67
6	12995	77
7	13035	57
8	13141	84
9	13183	7
10	13328	316
11	13581	309

페이지당 결과 수: 50 1 - 50 (전체 4362행)

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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_r AS
SELECT
  CustomerID,
  InvoiceDay,
  MAX(InvoiceDay) OVER () AS most_recent_date,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM project_name.modulabs_project.data
  WHERE UnitPrice > 0
  AND CustomerID IS NOT NULL
  GROUP BY CustomerID
);
```

행	CustomerID	InvoiceDay	most_recent_date	recency
1	16446	2011-12-09	2011-12-09	0
2	17428	2011-12-09	2011-12-09	0
3	14446	2011-12-09	2011-12-09	0
4	12423	2011-12-09	2011-12-09	0
5	17581	2011-12-09	2011-12-09	0
6	15344	2011-12-09	2011-12-09	0
7	15910	2011-12-09	2011-12-09	0
8	16705	2011-12-09	2011-12-09	0
9	15804	2011-12-09	2011-12-09	0
10	12433	2011-12-09	2011-12-09	0

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM project_name.modulabs_project.data
WHERE UnitPrice > 0
AND CustomerID IS NOT NULL
GROUP BY CustomerID;
```

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

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM project_name.modulabs_project.data
WHERE UnitPrice > 0
  AND CustomerID IS NOT NULL
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

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf AS
```

```
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM project_name.modulabs_project.data
  WHERE UnitPrice > 0
    AND CustomerID IS NOT NULL
  GROUP BY CustomerID
),
```

```
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM project_name.modulabs_project.data
  WHERE UnitPrice > 0
    AND CustomerID IS NOT NULL
  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 project_name.modulabs_project.user_r AS ur
ON pc.CustomerID = ur.CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	15520	1	314	1
3	14569	1	79	1
4	13436	1	76	1
5	13298	1	96	1
6	15195	1	1404	2
7	15471	1	256	2
8	14204	1	72	2
9	17914	1	457	3
10	16569	1	93	3
11	15318	1	642	3

페이지당 결과 수: 50 1 - 50 (전체 4362행)

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM project_name.modulabs_project.data
WHERE UnitPrice > 0
  AND CustomerID IS NOT NULL
GROUP BY CustomerID;
```

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

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 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 project_name.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 project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM project_name.modulabs_project.data
  WHERE UnitPrice > 0
  AND CustomerID IS NOT NULL
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;

```

일	CustomerID	purchase_amt	item_cnt	recency	user_stock	user_average
1	12713	1	505	0	794.5	794.5
2	13436	1	76	1	196.9	196.9
3	13298	1	96	1	360.0	360.0
4	15320	1	314	1	343.5	343.5
5	14569	1	79	1	227.4	227.4
6	14504	1	72	2	199.6	199.6
7	15195	1	1404	2	3801.0	3801.0
8	15471	1	256	2	454.5	454.5
9	16569	1	93	3	124.2	124.2
10	12442	1	181	3	144.1	144.1
11	12478	1	233	3	540.0	540.0
12	15992	1	17	3	42.0	42.0
13	16528	1	171	3	244.4	244.4

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```

SELECT *
FROM project_name.modulabs_project.user_rfm;

```

일	CustomerID	purchase_amt	item_cnt	recency	user_stock	user_average
1	12713	1	505	0	794.5	794.5
2	13436	1	76	1	196.9	196.9
3	13298	1	96	1	360.0	360.0
4	15320	1	314	1	343.5	343.5
5	14569	1	79	1	227.4	227.4
6	14504	1	72	2	199.6	199.6
7	15195	1	1404	2	3801.0	3801.0
8	15471	1	256	2	454.5	454.5
9	16569	1	93	3	124.2	124.2
10	12442	1	181	3	144.1	144.1
11	12478	1	233	3	540.0	540.0
12	15992	1	17	3	42.0	42.0
13	16528	1	171	3	244.4	244.4

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

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;

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15313	1	25	110	52.0	52.0	1
2	13302	1	5	155	63.8	63.8	1
3	14424	1	48	17	322.1	322.1	1
4	12791	1	96	373	177.6	177.6	1
5	16953	1	10	30	20.8	20.8	1
6	15889	1	72	217	76.3	76.3	1
7	16144	1	16	246	175.2	175.2	1
8	16078	1	16	283	79.2	79.2	1
9	12943	1	-1	301	-3.8	-3.8	1
10	13747	1	8	373	79.6	79.6	1
11	18184	1	60	15	49.8	49.8	1
12	17763	1	12	263	15.0	15.0	1
13	17956	1	1	249	12.8	12.8	1

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 군 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;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	14290	1	72	324	76.3	76.3	1	0.0
2	18184	1	60	15	49.8	49.8	1	0.0
3	17494	1	10	56	20.8	20.8	1	0.0
4	15389	1	430	172	500.0	500.0	1	0.0
5	15657	1	24	22	30.0	30.0	1	0.0
6	13391	1	4	203	59.8	59.8	1	0.0
7	15510	1	2	330	250.0	250.0	1	0.0
8	17443	1	504	219	534.2	534.2	1	0.0
9	15753	1	144	304	79.2	79.2	1	0.0
10	17357	1	144	305	152.6	152.6	1	0.0
11	16953	1	10	30	20.8	20.8	1	0.0
12	18113	1	72	368	76.3	76.3	1	0.0

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기
 - 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
 - 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 project_name.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 project_name.modulabs_project.data
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

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

일	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13327	1	4	120	150	150	1	0.0	1	0	0.0
2	13524	1	4	24	440.0	440.0	1	0.0	1	0	0.0
3	14139	1	28	218	185.4	185.4	2	0.0	1	0	0.0
4	17610	1	216	266	284.2	284.2	5	0.0	1	0	0.0
5	12403	1	94	49	391.7	391.7	5	0.0	1	0	0.0
6	13845	1	158	120	252.1	252.1	5	0.0	1	0	0.0
7	17844	1	82	106	81.6	81.6	5	0.0	1	0	0.0
8	12659	1	104	29	73.7	73.7	5	0.0	1	0	0.0
9	14542	1	19	185	103.2	103.2	5	0.0	1	0	0.0
10	15775	1	68	245	104.9	104.9	7	0.0	1	0	0.0
11	13876	1	36	63	122.7	122.7	7	0.0	1	0	0.0
12	19158	1	33	74	104.3	104.3	7	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 project_name.modulabs_project.user_data
ORDER BY recency ASC
LIMIT 100;
```

일	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	11982	11	822	0	1938.9	176.1	63	2.08	11	4	0.36
2	12519	5	1356	0	1840.9	368.2	83	0.92	5	0	0.0
3	16626	20	2670	0	4379.7	219.0	86	1.4	20	3	0.15
4	15910	8	1013	0	1228.9	153.6	215	1.18	8	0	0.0
5	13777	40	12807	0	25788.1	644.0	39	1.85	40	7	0.17
6	13426	11	2225	0	3858.9	352.5	109	2.25	11	1	0.09
7	12612	12	2023	0	9511.1	292.6	128	1.67	12	1	0.08
8	18102	60	64124	0	299457.3	4327.6	150	0.82	60	0	0.0
9	14422	6	2906	0	4265.6	710.6	189	1.19	6	0	0.0
10	14481	10	430	0	1545.1	154.5	45	6.19	10	6	0.6
11	15994	14	1864	0	6459.6	457.8	43	4.76	14	1	0.07
12	12526	3	624	0	1172.7	390.9	63	1.36	3	0	0.0
13	16795	79	6406	0	17944.1	440.9	135	1.79	79	6	0.71

회고

[회고 내용을 작성해주세요]

Keep :

- 단계별로 데이터를 정제하고, 불필요한 값을 제거한 과정
- Recency, Frequency, Monetary를 포함한 고객별 핵심 Feature를 체계적으로 계산하고 통합한 점
- 추가 Feature를 통해 고객 구매 패턴과 선호도를 심층적으로 이해할 수 있도록 확장한 점
- 서브쿼리, 윈도우 함수, 집계 함수 등 SQL기능을 적절히 활용하여 복잡한 계산을 효율적으로 처리한 점

Problem :

- 일부 쿼리에서 집계 함수와 일반 컬럼 혼합으로 오류가 발생(max() + * 문제)
- 날짜 차이를 계산할 때 DATE_DIFF와 EXTRACT 사용 방식이 혼동되어 잠시 시행착오 발생
- 여러 Feature를 통합할 때 컬럼 이름 중복 문제 발생

Try :

- 집계 함수와 일반 컬럼을 사용할 때는 윈도우 함수 또는 서브쿼리를 적극 활용하여 오류 방지
- 추가 Feature를 설계할 때는 단계별로 CTE를 활용하여 가독성과 재사용성을 높이기
- 추후 분석에서는 고객 세그먼트별 통계 요약이나 시각화까지 연결하여 분석 결과를 직관적으로 확인