

고객을 세그멘테이션하자 [프로젝트] (1)

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM velvety-study-473605-t0.modulabs_project.data  
LIMIT 10
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
1	536365	85123A	WHITE HANGING HEART T.LIG...	6	2010-12-01 08:26:00 UTC	2.55	178
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	178
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	178
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	178
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	178
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	178
7	536365	21730	GLASS STAR FROSTED T.LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	178
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	178
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	178
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	180

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)  
FROM velvety-study-473605-t0.modulabs_project.data
```

행	f0_
1	541909

데이터 수 세기

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

```
SELECT  
COUNT(InvoiceNo) AS count_InvoiceNo,  
COUNT(StockCode) AS count_StockCode,  
COUNT(Description) AS count_Description,  
COUNT(Quantity) AS count_Quantity,  
COUNT(InvoiceDate) AS count_InvoiceDate,  
COUNT(UnitPrice) AS count_UnitPrice,  
COUNT(CustomerID) AS count_CustomerID,  
COUNT(Country) AS count_Country  
FROM velvety-study-473605-t0.modulabs_project.data
```

행	count_InvoiceNo	count_StockCode	count_Description	count_Quantity	count_InvoiceDate	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) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data
UNION ALL
SELECT
    'StockCode' AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data
UNION ALL
SELECT
    'Description' AS column_name,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data
UNION ALL
SELECT
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data
UNION ALL
SELECT
    'InvoiceDate' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data
UNION ALL
SELECT
    'UnitPrice' AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data
UNION ALL
SELECT
    'CustomerID' AS column_name,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data
UNION ALL
SELECT
    'Country' AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) * 100 / COUNT(*), 2) AS missing_percentage
FROM
    velvety-study-473605-t0.modulabs_project.data

```

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM velvety-study-473605-t0.modulabs_project.data
WHERE StockCode = '85123A'
```

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 velvety-study-473605-t0.modulabs_project.data
WHERE CustomerID IS NULL OR Description IS NULL;
```

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT
  COUNT(*) AS num_duplicate_groups
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  FROM
    velvety-study-473605-t0.modulabs_project.data
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
```

```
Quantity,
InvoiceDate,
UnitPrice,
CustomerID,
Country
HAVING
COUNT(*) > 1
```

← 쿼리 결과

작업 정보	결과	시각화
행	num_duplicate_gr...	
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 velvety-study-473605-t0.modulabs_project.data_cleaned AS
SELECT DISTINCT *
FROM velvety-study-473605-t0.modulabs_project.data
WHERE
  CustomerID IS NOT NULL
  AND Description IS NOT NULL;
```

(실행두번눌러서 data_cleaned로 나오네요..)

i 이 문으로 이름이 data_cleaned인 테이블이 교체되었습니다.

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM velvety-study-473605-t0.modulabs_project.data_cleaned;
```

작업 정보	결과	시각화
행	unique_invoice_c...	
1	22190	

- 고유한 InvoiceNo를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo AS unique_invoice
FROM velvety-study-473605-t0.modulabs_project.data_cleaned
LIMIT 100;
```

행	unique_invoice
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318
11	541998

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

```
SELECT *
FROM velvety-study-473605-t0.modulabs_project.data_cleaned
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custom
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	
5	C547388	22645	CERAMIC HEART FAIRY CAKE...	-12	2011-03-22 16:07:00 UTC	1.45	
6	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	

페이지당 결과 수: 50 1 ~ 50 (전체 100행) |< > >|

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) * 100.0 / COUNT(*),1) AS cancellation_row_rate_percentage
FROM velvety-study-473605-t0.modulabs_project.data_cleaned;
```

행	cancellation_row...
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockCode
FROM velvety-study-473605-t0.modulabs_project.data_cleaned
```

행	unique_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 velvety-study-473605-t0.modulabs_project.data_cleaned
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

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

- 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 velvety-study-473605-t0.modulabs_project.data_cleaned
)
WHERE number_count IN (0,1)
```

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
  DISTINCT StockCode,
  number_count
FROM
  (SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM
    velvety-study-473605-t0.modulabs_project.data_cleaned
  )
WHERE number_count IN (0, 1);
```

행	outlier_percentage
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM velvety-study-473605-t0.modulabs_project.data_cleaned
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (SELECT StockCode,LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM velvety-study-473605-t0.modulabs_project.data_cleaned
  )
  WHERE number_count IN (0, 1)
);
```

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data_cleaned의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description,COUNT(*) AS description_cnt
FROM velvety-study-473605-t0.modulabs_project.data_cleaned
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

행	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY D...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE ...	1062
10	SPOTTY BUNTING	1026

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM velvety-study-473605-t0.modulabs_project.data_cleaned
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data_cleaned의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs_project.data_standardized AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM
  velvety-study-473605-t0.modulabs_project.data_cleaned;
```

i 이 문으로 이름이 data_standardized인 새 테이블이 생성되었습니다.

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
```



```
AVG(UnitPrice) AS avg_price
FROM
velvety-study-473605-t0.modulabs_project.data_standardized;
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

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

```
SESELECT
COUNT(*) AS cnt_quantity,
MIN(Quantity) AS min_quantity,
MAX(Quantity) AS max_quantity,
AVG(Quantity) AS avg_quantity
FROM
velvety-study-473605-t0.modulabs_project.data_standardized
WHERE
UnitPrice = 0;
```

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
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 velvety-study-473605-t0.modulabs_project.data AS
SELECT *
FROM velvety-study-473605-t0.modulabs_project.data_standardized
WHERE UnitPrice > 0;
```

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay,*
FROM velvety-study-473605-t0.modulabs_project.data;
```

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프		
행	InvoiceDay ▾	InvoiceNo ▾	StockCode ▾	Quantity ▾	InvoiceDate ▾	UnitPrice ▾	CustomerID ▾
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346
2	2011-01-18	0541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346
3	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347
4	2010-12-07	537626	22729	4	2010-12-07 14:57:00 UTC	3.75	12347
5	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC	3.75	12347
6	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	12347

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,*
FROM velvety-study-473605-t0.modulabs_project.data;
```

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-12-09	2011-10-13	571034	23094	4	2011-10-13 12:47:3
2	2011-12-09	2011-08-12	563100	23163	8	2011-08-12 09:57:3
3	2011-12-09	2011-05-17	553546	23299	50	2011-05-17 15:42:1
4	2011-12-09	2011-11-15	576394	22196	160	2011-11-15 10:32:1
5	2011-12-09	2011-02-09	543541	21317	2	2011-02-09 14:44:1
6	2011-12-09	2011-02-03	543117	22847	4	2011-02-03 13:30:1
7	2011-12-09	2010-12-10	538174	22423	32	2010-12-10 09:35:1
8	2011-12-09	2011-03-18	547005	23007	1	2011-03-18 14:20:1
9	2011-12-09	2011-11-14	576327	23531	3	2011-11-14 15:19:1
10	2011-12-09	2011-11-17	576910	22191	12	2011-11-17 09:51:1
11	2011-12-09	2011-09-19	567291	23319	6	2011-09-19 11:55:1

페이지 12 / 200 개

1 ~ 50 / 전체 200 개

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM velvety-study-473605-t0.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

- 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 velvety-study-473605-t0.modulabs_project.data
  GROUP BY CustomerID
);

```

행	CustomerID	recency
1	12539	22
2	12579	73
3	12604	79
4	12621	1
5	12625	211
6	12865	26
7	12977	156

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

```

CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs_project.user_r AS
SELECT
  CustomerID,
  DATE_DIFF(MAX(InvoiceDay) OVER (), InvoiceDay, DAY) AS Recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM
    velvety-study-473605-t0.modulabs_project.data
  GROUP BY
    CustomerID
);

```

i 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```

SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM
  velvety-study-473605-t0.modulabs_project.data
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

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM
  velvety-study-473605-t0.modulabs_project.data
WHERE
  Quantity > 0
GROUP BY
  CustomerID;
```

행	CustomerID	item_cnt
1	12346	74215
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	526
7	12353	20
8	12354	530
9	12355	240

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

```
CREATE OR REPLACE TABLE velvety-study-473605-t0.modulabs_project.user_rf AS
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM
    velvety-study-473605-t0.modulabs_project.data
  GROUP BY
```

```

CustomerID
),

item_cnt AS (
SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
FROM
    velvety-study-473605-t0.modulabs_project.data
WHERE
    Quantity > 0
GROUP BY
    CustomerID
)

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 velvety-study-473605-t0.modulabs_project.user_r AS ur
    ON pc.CustomerID = ur.CustomerID;

SELECT *
FROM velvety-study-473605-t0.modulabs_project.user_rf

```

행	CustomerID	purchase_cnt	item_cnt	Recency
1	15165	1	160	373
2	13747	1	8	373
3	16583	1	111	373
4	17908	1	169	373
5	13065	1	74	373
6	17643	1	71	373
7	14237	1	38	373
8	12791	1	96	373
9	18074	1	190	373

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 0) AS user_total
FROM
    velvety-study-473605-t0.modulabs_project.data
GROUP BY
    CustomerID
HAVING
    SUM(Quantity * UnitPrice) > 0;

```

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세
행	CustomerID	user_total		
1	12347	4310.0		
2	12348	1437.0		
3	12349	1458.0		
4	12350	294.0		
5	12352	1265.0		
6	12353	89.0		
7	12354	1079.0		
8	12355	459.0		
9	12356	2487.0		
10	12357	6208.0		

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 velvety-study-473605-t0.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.Recency,
  ut.user_total,
  ROUND(ut.user_total / rf.purchase_cnt, 2) AS user_average
FROM
  velvety-study-473605-t0.modulabs_project.user_rf AS rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 0) AS user_total
  FROM
    velvety-study-473605-t0.modulabs_project.data
  GROUP BY
    CustomerID
  HAVING
    SUM(Quantity * UnitPrice) > 0
) AS ut
ON
  rf.CustomerID = ut.CustomerID;
```

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM velvety-study-473605-t0.modulabs_project.user_rfm
LIMIT 10;
```

← 쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

명	CustomerID	purchase_cnt	item_cnt	Recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	15520	1	314	1	343.0	343.0
3	14569	1	79	1	227.0	227.0
4	13298	1	96	1	360.0	360.0
5	13436	1	76	1	197.0	197.0
6	15471	1	256	2	454.0	454.0
7	15195	1	1404	2	3861.0	3861.0
8	14204	1	72	2	151.0	151.0
9	17914	1	457	3	329.0	329.0
10	15992	1	17	3	42.0	42.0

페이지당 결과 수: 50

1 - 10 (전체 10행)

<

>

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 velvety-study-473605-t0.modulabs_project.user_data AS
WITH
  unique_products AS (
    SELECT
      CustomerID,
      COUNT(DISTINCT StockCode) AS unique_products
    FROM
      velvety-study-473605-t0.modulabs_project.data
    GROUP BY
      CustomerID
  )
SELECT
  ur.*,
  up.unique_products
FROM
  velvety-study-473605-t0.modulabs_project.user_rfm AS ur
JOIN
  unique_products AS up
  ON ur.CustomerID = up.CustomerID;

SELECT *
FROM velvety-study-473605-t0.modulabs_project.user_data
```

작업 정보 **결과** 시각화 JSON 실행 세부정보 실행 그래프

명	CustomerID ▾	purchase_cnt ▾	item_cnt ▾	Recency ▾	user_total ▾	user_average ▾	unique_products ▾	
1	12713	1	505	0	795.0	795.0	37	
2	14569	1	79	1	227.0	227.0	10	
3	13436	1	76	1	197.0	197.0	12	
4	13298	1	96	1	360.0	360.0	2	
5	15520	1	314	1	343.0	343.0	18	
6	14204	1	72	2	151.0	151.0	36	
7	15195	1	1404	2	3861.0	3861.0	1	
8	15471	1	256	2	454.0	454.0	67	
9	15318	1	642	3	313.0	313.0	33	
10	15992	1	17	3	42.0	42.0	3	

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 velvety-study-473605-t0.modulabs_project.user_data AS
WITH
  purchase_intervals AS (
    SELECT
      CustomerID,
      COALESCE(ROUND(AVG(interval_days), 2), 0) AS average_interval
    FROM (
      SELECT
        CustomerID,
        DATE_DIFF(
          DATE(InvoiceDate),
          LAG(DATE(InvoiceDate)) OVER (PARTITION BY CustomerID ORDER BY DATE(InvoiceDate)),
          DAY
        ) AS interval_days
      FROM
        velvety-study-473605-t0.modulabs_project.data
    )
    GROUP BY
      CustomerID
  )

SELECT
  u.* EXCEPT (average_interval),
  pi.average_interval
FROM
  velvety-study-473605-t0.modulabs_project.user_data AS u
LEFT JOIN
  purchase_intervals AS pi
  ON u.CustomerID = pi.CustomerID;

SELECT *
FROM velvety-study-473605-t0.modulabs_project.user_data
```

작업 정보		결과	시각화	JSON	실행 세부정보	실행 그래프			
seq	CustomerID ▾	purchase_cnt ▾	item_cnt ▾	Recency ▾	user_total ▾	user_average ▾	unique_products ▾	average_interval ▾	
1	16881	1	600	66	432.0	432.0	1	0.0	
2	13302	1	5	155	64.0	64.0	1	0.0	
3	16344	1	18	158	101.0	101.0	1	0.0	
4	17763	1	12	263	15.0	15.0	1	0.0	
5	16737	1	288	53	418.0	418.0	1	0.0	
6	15195	1	1404	2	3861.0	3861.0	1	0.0	
7	16990	1	100	218	179.0	179.0	1	0.0	
8	13307	1	4	120	15.0	15.0	1	0.0	
9	14424	1	48	17	322.0	322.0	1	0.0	
10	13135	1	4300	196	3096.0	3096.0	1	0.0	

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 velvety-study-473605-t0.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
      velvety-study-473605-t0.modulabs_project.data
    GROUP BY
      CustomerID
  )
SELECT
  u.CustomerID,
  u.purchase_cnt,
  u.item_cnt,
  u.Recency,
  u.user_total,
  u.user_average,
  u.unique_products,
  u.average_interval,
  COALESCE(t.cancel_frequency, 0) AS cancel_frequency,
  COALESCE(
    ROUND((t.cancel_frequency / t.total_transactions) * 100, 2),
    0
  ) AS cancel_rate
FROM
  velvety-study-473605-t0.modulabs_project.user_data AS u
LEFT JOIN
  TransactionInfo AS t
ON u.CustomerID = t.CustomerID;

```

작업 정보 **결과** 실행 세부정보 실행 그래프

i 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 velvety-study-473605-t0.modulabs_project.user_data

```

←

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

customerID	purchase_cnt	item_cnt	Recency	user_total	user_average	unique_products	average_interval	cancel_frequency	cancel_rate
17763	1	12	263	15.0	15.0	1	0.0	0	0
15753	1	144	304	79.0	79.0	1	0.0	0	0
16323	1	50	196	208.0	208.0	1	0.0	0	0
12917	1	120	128	594.0	594.0	2	0.0	0	0
16319	1	72	217	404.0	404.0	3	0.0	0	0
14457	1	88	310	114.0	114.0	4	0.0	0	0
14582	1	73	40	114.0	114.0	6	0.0	0	0
16114	1	43	170	136.0	136.0	8	0.0	0	0
18280	1	45	277	181.0	181.0	10	0.0	0	0

회고

Keep : 중간중간 잘 돌아가는지 확인하여서 오류가 생기면 그때그때 확인하고 고칠 수 있었던 점이 좋았습니다. 또한 흐름에 맞춰 코드 중간중간 구간을 표시하여 코드를 되돌아가야하는 때가 생겼을때 빠르게 다시 찾을 수 있었습니다. 그 점이 좋았습니다.

Problem : 쿼리 결과에서 페이지를 넘길 수 있다는 사실을 몰라서

답지와 고객 수, recency값이 다르게 나온것처럼 보여(행이 순서대로 나오지 않는다는 점을 간과하였음) 원인을 찾는 데 많은 시간이 소요되었습니다..

Try : 다음 프로젝트에서는 분석 목표에 따른 데이터 정제 규칙을 먼저 명확히 정의하여, 정답지와의 결과 차이로 인한 혼란을 줄이겠습니다.