

고객을 세그멘테이션하자 [프로젝트] (1)

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
select *  
from infinite-chain-456201-u5.modulabs_project.data  
limit 10
```

[결과 이미지를 넣어주세요]

쿼리 결과						
<div>결과 저장 다음에서 열기</div>						
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프	
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC	
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-12-01 08:26:00 UTC	
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC	
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
select count(*)  
from infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과		
<div>결과 차트 JSON 실행 세부정보 실행 그래프</div>		
작업 정보	결과	차트
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 infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID
1	541909	541909	540455	541909	541909	541909	406829

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 *
FROM(
  (SELECT 'InvoiceNo'AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data) UNION ALL
  (SELECT 'StockCode'AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data) UNION ALL
  (SELECT 'Description'AS column_name,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data) UNION ALL
  (SELECT 'Quantity'AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data) UNION ALL
  (SELECT 'InvoiceDate'AS column_name,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data) UNION ALL
  (SELECT 'UnitPrice'AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data) UNION ALL
  (SELECT 'CustomerID'AS column_name,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data) UNION ALL
  (SELECT 'Country'AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM infinite-chain-456201-u5.modulabs_project.data)
)
ORDER BY 2 DESC
```

[결과 이미지를 넣어주세요]

작업 정보 결과 차트 JSON 실행 세

행	column_name ▼	missing_percentage
1	CustomerID	24.93
2	Description	0.27
3	InvoiceDate	0.0
4	Quantity	0.0
5	Country	0.0

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

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 infinite-chain-456201-u5.modulabs_project.data
WHERE CustomerID IS NULL
OR Description IS NULL
```

[결과 이미지를 넣어주세요]

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, COUNT가 1보다 큰 데이터를 세어보기

```
SELECT *
FROM infinite-chain-456201-u5.modulabs_project.data
GROUP BY InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
HAVING COUNT(*) > 1
```

[결과 이미지를 넣어주세요]

- 실수로 중값값 처리 이전 단계의 캡처본(중복 4837행)을 날려버렸고, 중복값 제거 이후에 실행 시 다음처럼 아무 것도 안 뜹니다. (중복 데이터 없음)

← 쿼리 결과

작업 정보 **결과** 차트 JSON 실행 세부정보

i 표시할 데이터가 없습니다.

중복값 처리

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

```
CREATE OR REPLACE TABLE infinite-chain-456201-u5.modulabs_project.data AS
SELECT DISTINCT *
FROM infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(distinct InvoiceNo)
FROM infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

행	f0_
1	22190

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT distinct InvoiceNo
FROM infinite-chain-456201-u5.modulabs_project.data
limit 100
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
1	541431				
2	C541433				
3	537626				
4	542237				
5	549222				

페이지당 결과 수: 50 1 - 50 (전체 100행)

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

```
SELECT *
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE InvoiceNo like "C%"
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 U
2	C545329	M	Manual	-1	2011-03-01 15:47:00 U
3	C545329	M	Manual	-1	2011-03-01 15:47:00 U
4	C545330	M	Manual	-1	2011-03-01 15:49:00 U
5	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 U

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) / count(*) * 100, 1)
FROM infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과
행	f0_
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과
행	f0_
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 infinite-chain-456201-u5.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	StockCode	sell_cnt			
1	85123A	2065			
2	22423	1894			
3	85099B	1659			
4	47566	1409			
5	84879	1405			

페이지당 결과 수: 50 1 - 10 (전체 10행)

- **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 infinite-chain-456201-u5.modulabs_project.data
)
WHERE number_count < 5;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보			결과	차트	JSON	실행 세
행	StockCode	number_count				
1	POST	0				
2	M	0				
3	C2	1				
4	D	0				
5	BANK CHARGES	0				
6	PADS	0				
7	DOT	0				
8	CRUK	0				

- **StockCode**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
SELECT ROUND(SUM(CASE WHEN StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK') THEN 1 ELSE 0)) / COUNT(*)
FROM infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보		결과
행	f0_	
1		0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM infinite-chain-456201-u5.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT DISTINCT StockCode, number_count
    FROM (
      SELECT StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
      FROM infinite-chain-456201-u5.modulabs_project.data
    )
    WHERE number_count < 5
  )
)
```

```
)  
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 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 infinite-chain-456201-u5.modulabs_project.data  
GROUP BY Description  
ORDER BY 2 DESC  
LIMIT 30;
```

[결과 이미지를 넣어주세요]

쿼리 결과

[결과 저장](#)

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
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			

페이지당 결과 수: 50 ▼ 1 - 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE  
FROM infinite-chain-456201-u5.modulabs_project.data  
WHERE Description IN (  
  SELECT DISTINCT Description  
  FROM infinite-chain-456201-u5.modulabs_project.data  
  WHERE REGEXP_CONTAINS(Description, r'[a-z]')  
  AND Description IN ("High Resolution Image","Next Day Carriage")  
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE infinite-chain-456201-u5.modulabs_project.data AS  
SELECT
```

```
* EXCEPT (Description),
UPPER(Description) AS Description
FROM infinite-chain-456201-u5.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 data인 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 infinite-chain-456201-u5.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 차트 JSON 실행 세부정보

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

```
SELECT count(Quantity) AS cnt_quantity,
       min(Quantity) AS min_quantity,
       max(Quantity) AS max_quantity,
       avg(Quantity) AS avg_quantity
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

작업 정보 **결과** 차트 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 infinite-chain-456201-u5.modulabs_project.data AS (
SELECT *
FROM infinite-chain-456201-u5.modulabs_project.data
WHERE UnitPrice <> 0
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *  
FROM infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC
3	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC
4	2010-12-07	537626	71477	12	2010-12-07 14:57:00 UTC
5	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC

- 가장 최근 구매 일자를 **MAX()** 함수로 찾아보기

```
SELECT  
  MAX(DATE(InvoiceDate)) AS most_recent_date  
FROM infinite-chain-456201-u5.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

행	most_recent_date
1	2011-12-09

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT  
  CustomerID,  
  DATE(MAX(InvoiceDate)) AS InvoiceDay  
FROM infinite-chain-456201-u5.modulabs_project.data  
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON
행	CustomerID	InvoiceDay	
1	12346	2011-01-18	
2	12347	2011-12-07	
3	12348	2011-09-25	
4	12349	2011-11-21	
5	12350	2011-02-02	

- 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 infinite-chain-456201-u5.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	CustomerID	recency
1	12450	156
2	12457	58
3	12472	30
4	12691	28
5	13435	5

페이지당 결과 수: 50 1 - 50 (전체 4362행)

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,
    EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
  FROM (
    SELECT
      CustomerID,
      MAX(DATE(InvoiceDate)) AS InvoiceDay
    FROM infinite-chain-456201-u5.modulabs_project.data
    GROUP BY CustomerID
  )
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프

i 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM infinite-chain-456201-u5.modulabs_project.data
GROUP BY 1
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	
작업 정보	결과	차트	JSON	실행 세부정보
실행 그래프				
행	CustomerID	purchase_cnt		
1	12346	2		
2	12347	182		
3	12348	27		
4	12349	72		
5	12350	16		
			페이지당 결과 수:	50 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT CustomerID,
  SUM(Quantity) AS item_cnt
FROM infinite-chain-456201-u5.modulabs_project.data
GROUP BY 1
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	
작업 정보	결과	차트	JSON	실행 세부정보
실행 그래프				
행	CustomerID	item_cnt		
1	12346	0		
2	12347	2458		
3	12348	2332		
4	12349	630		
5	12350	196		
			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 infinite-chain-456201-u5.modulabs_project.user_rf AS (
  -- (1) 전체 거래 건수 계산
  WITH purchase_cnt AS (
    SELECT
      CustomerID,
      COUNT(DISTINCT InvoiceNo) AS purchase_cnt
    FROM infinite-chain-456201-u5.modulabs_project.data
    GROUP BY 1
```

```

),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT CustomerID,
    SUM(Quantity) AS item_cnt
  FROM infinite-chain-456201-u5.modulabs_project.data
  GROUP BY 1
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.reccency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN infinite-chain-456201-u5.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID
)

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
  CustomerID,
  ROUND(sum(UnitPrice * Quantity),1) AS user_total
FROM infinite-chain-456201-u5.modulabs_project.data
GROUP BY 1

```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	user_total			
1	12346	0.0			
2	12347	4310.0			
3	12348	1437.2			
4	12349	1457.6			
5	12350	294.4			

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 infinite-chain-456201-u5.modulabs_project.user_rfm AS (
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ut.user_total / rf.purchase_cnt AS user_average
FROM infinite-chain-456201-u5.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(sum(UnitPrice * Quantity),1) AS user_total
  FROM infinite-chain-456201-u5.modulabs_project.data
  GROUP BY 1
) ut
ON rf.CustomerID = ut.CustomerID
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

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 infinite-chain-456201-u5.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

쿼리 결과

[결과 저장](#) [다음에](#)

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	794.5
2	15520	1	314	1	343.5	343.5
3	13436	1	76	1	196.9	196.9
4	14569	1	79	1	227.4	227.4
5	13298	1	96	1	360.0	360.0

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 infinite-chain-456201-u5.modulabs_project.user_data AS (
  WITH unique_products AS (
    SELECT
      CustomerID,
      COUNT(DISTINCT StockCode) AS unique_products
    FROM infinite-chain-456201-u5.modulabs_project.data
    GROUP BY CustomerID
  )
  SELECT ur.*, up.* EXCEPT (CustomerID)
  FROM infinite-chain-456201-u5.modulabs_project.user_rfm AS ur
  JOIN unique_products AS up
  ON ur.CustomerID = up.CustomerID
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

 이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

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 infinite-chain-456201-u5.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
        infinite-chain-456201-u5.modulabs_project.data
      WHERE CustomerID IS NOT NULL
    )
    GROUP BY CustomerID
  )
  SELECT u.*, pi.* EXCEPT (CustomerID)
  FROM infinite-chain-456201-u5.modulabs_project.user_data AS u
  LEFT JOIN purchase_intervals AS pi
  ON u.CustomerID = pi.CustomerID
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프

i 이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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 project_name.modulabs_project.user_data AS (

WITH TransactionInfo AS (
    SELECT
        CustomerID,
        count(InvoiceNo) AS total_transactions,
        SUM(CASE WHEN InvoiceNo like "C%" THEN 1 ELSE 0 END) AS cancel_frequency
    FROM infinite-chain-456201-u5.modulabs_project.data
    GROUP BY 1
)

SELECT u.*, t.* EXCEPT(CustomerID), round(t.cancel_frequency / t.total_transactions * 100, 2) AS cancel_rate
FROM infinite-chain-456201-u5.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프

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 infinite-chain-456201-u5.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기



작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15870	1	333	32	351.4	351.4	68
2	14675	1	336	16	596.4	596.4	93
3	16795	1	239	365	414.9	414.9	64
4	17379	1	623	11	389.7	389.7	71
5	12965	1	282	89	769.8	769.8	108

페이지당 결과 수: 50

1 - 50 (전체 4362행)

이전 다음

회고

[회고 내용을 작성해주세요]

Keep : 간신히 완성했다.

Problem : 해석에 대해 깊게 고려해볼 시간이 부족하다.

Try : 결과물에 대한 단계별 회고가 필요하다.