

고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM angelic-goods-456101-n9.modulabs_project.data  
LIMIT 10;
```

[결과 이미지를 넣어주세요]

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

페이지당 결과 수: 50 1 - 10 (전체 10행) < >

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT *  
FROM angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

페이지당 결과 수: 50 1 - 50 (전체 541909행) < >

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 angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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) /
  COUNT(*) * 100, 2) AS missing_percentage
FROM angelic-goods-456101-n9.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 angelic-goods-456101-n9.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 angelic-goods-456101-n9.modulabs_project.data

UNION ALL

SELECT
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / C
OUNT(*) * 100, 2) AS missing_percentage
FROM angelic-goods-456101-n9.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 angelic-goods-456101-n9.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 angelic-goods-456101-n9.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 angelic-goods-456101-n9.modulabs_project.data
```

```
UNION ALL
```

```
SELECT
  'Country' AS column_name,
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / C
OUNT(*) * 100, 2) AS missing_percentage
FROM angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

결측치 처리 전략

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

```
SELECT Description
FROM angelic-goods-456101-n9.modulabs_project.data
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

행	Description ▼
1	WHITE HANGING HEART T-LIG...
2	WHITE HANGING HEART T-LIG...
3	WHITE HANGING HEART T-LIG...
4	WHITE HANGING HEART T-LIG...
5	WHITE HANGING HEART T-LIG...
6	WHITE HANGING HEART T-LIG...
7	WHITE HANGING HEART T-LIG...

결측치 처리

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

```
-- CustomerID가 NULL인 행 제거
DELETE FROM angelic-goods-456101-n9.modulabs_project.data
WHERE CustomerID IS NULL;

-- Description이 NULL인 행 제거
DELETE FROM angelic-goods-456101-n9.modulabs_project.data
WHERE Description IS NULL;
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 135,080개가 삭제되었습니다.

i 이 문으로 data의 행 0개가 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_row_count
FROM (
    SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
    FROM angelic-goods-456101-n9.modulabs_project.data
    GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
    HAVING COUNT(*) > 1
) AS duplicates;
```

[결과 이미지를 넣어주세요]

행	duplicate_row_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 angelic-goods-456101-n9.modulabs_project.data AS
SELECT DISTINCT *
FROM angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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) AS unique_invoice_count  
FROM angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	unique_invoice_coun
1	22190

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo  
FROM angelic-goods-456101-n9.modulabs_project.data  
ORDER BY InvoiceNo  
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo
1	536365
2	536366
3	536367
4	536368
5	536369
6	536370

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

```
SELECT *
FROM angelic-goods-456101-n9.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quan
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	
2	C545329	M	Manual	
3	C545329	M	Manual	
4	C545330	M	Manual	
5	C547388	84050	PINK HEART SHAPE EGG FRYI...	
6	C547388	22645	CERAMIC HEART FAIRY CAKE ...	
7	C547388	22784	LANTERN CREAM GAZEBO	
8	C547388	37448	CERAMIC CAKE DESIGN SPOT ...	
9	C547388	21914	BLUE HARMONICA IN BOX	
10	C547388	22413	METAL SIGN TAKE IT OR LEAV...	
11	C547388	22701	PINK DOG BOWL	
12	C549955	22839	3 TIER CAKE TIN GREEN AND ...	
13	C549955	22666	RECIPE BOX PANTRY YELLOW ...	

- 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 angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	canceled_percentage
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM angelic-goods-456101-n9.modulabs_project.data;
```


[결과 이미지를 넣어주세요]

행	unique_stockcode_cn
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 angelic-goods-456101-n9.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

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

- StockCode**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 숫자가 0~1개인 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 angelic-goods-456101-n9.modulabs_project.data
)
WHERE number_count < 5;
```

[결과 이미지를 넣어주세요]

행	StockCode	number_count
1	POST	0
2	M	0
3	C2	1
4	D	0
5	BANK CHARGES	0
6	PADS	0
7	DOT	0
8	CRUK	0

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

```
SELECT ROUND(SUM(CASE WHEN number_count < 5 THEN 1 ELSE 0 E
ND) / COUNT(*) * 100, 2)
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
```

```
FROM angelic-goods-456101-n9.modulabs_project.data
);
```


[결과 이미지를 넣어주세요]

행	low_digit_stockcode
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,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM angelic-goods-456101-n9.modulabs_project.data
  )
  WHERE number_count < 5
);
```

[결과 이미지를 넣어주세요]

 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 angelic-goods-456101-n9.modulabs_project.data
GROUP BY Description
```

```
ORDER BY COUNT(*) DESC
LIMIT 30;
```


[결과 이미지를 넣어주세요]

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 ...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE...	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM angelic-goods-456101-n9.modulabs_project.data
WHERE
  LOWER(Description) LIKE '%service%' OR
  LOWER(Description) LIKE '%charges%' OR
  LOWER(Description) LIKE '%carriage%' OR
  LOWER(Description) LIKE '%manual%' OR
  LOWER(Description) LIKE '%adjustment%' OR
  LOWER(Description) LIKE '%surcharge%' OR
  LOWER(Description) LIKE '%check%' OR
  LOWER(Description) LIKE '%discount%';
```


[결과 이미지를 넣어주세요]

 이 문으로 data의 행 301개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE angelic-goods-456101-n9.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

 이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	min_price	max_price	avg_price
1	0.0	649.5	2.9

- 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 angelic-goods-456101-n9.modulabs_project.data
WHERE UnitPrice = 0;
```


[결과 이미지를 넣어주세요]

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.52

- **UnitPrice = 0** 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT *
FROM angelic-goods-456101-n9.modulabs_project.data
WHERE UnitPrice > 0;
```

[결과 이미지를 넣어주세요]

 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	InvoiceDay ▼	InvoiceNo ▼	StockCode ▼	Quantity ▼
1	2011-01-18	541431	23166	74215
2	2011-01-18	C541433	23166	-74215
3	2010-12-07	537626	84969	6
4	2010-12-07	537626	22195	12
5	2010-12-07	537626	22492	36
6	2010-12-07	537626	21171	12
7	2010-12-07	537626	20780	12
8	2010-12-07	537626	22375	4

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM angelic-goods-456101-n9.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	most_recent_date ▼	InvoiceDay ▼	InvoiceNo ▼	StockCode ▼	Quantity ▼	InvoiceDate ▼
1	2011-12-09	2011-03-10	546236	84598	288	2011-03-10 12:48:00 UTC
2	2011-12-09	2011-02-07	543357	47590B	3	2011-02-07 13:59:00 UTC
3	2011-12-09	2011-01-11	540700	21581	6	2011-01-11 09:47:00 UTC
4	2011-12-09	2011-02-15	543989	22716	144	2011-02-15 09:52:00 UTC
5	2011-12-09	2011-06-15	556917	85099B	100	2011-06-15 13:37:00 UTC
6	2011-12-09	2011-10-05	569650	23525	48	2011-10-05 12:44:00 UTC
7	2011-12-09	2011-11-04	574469	23084	336	2011-11-04 11:55:00 UTC
8	2011-12-09	2011-01-12	540972	22236	12	2011-01-12 14:13:00 UTC
9	2011-12-09	2011-01-19	541518	22847	4	2011-01-19 09:05:00 UTC
10	2011-12-09	2011-01-07	540414	21883	24	2011-01-07 10:32:00 UTC
11	2011-12-09	2011-10-11	570672	84929	24	2011-10-11 14:52:00 UTC
12	2011-12-09	2011-10-11	570672	21055	1	2011-10-11 14:52:00 UTC
13	2011-12-09	2011-11-04	574501	21371	3	2011-11-04 13:15:00 UTC

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM angelic-goods-456101-n9.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID ▼	InvoiceDay ▼
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17
11	12357	2011-11-06
12	12358	2011-12-08
13	12359	2011-12-02
14	12360	2011-10-18

- 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 angelic-goods-456101-n9.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	12407	49
2	12489	336
3	12577	35
4	12578	21
5	12581	39
6	12684	7
7	12712	22
8	12715	106
9	12818	178
10	12847	22
11	13052	212
12	13138	22
13	13475	191

- 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 angelic-goods-456101-n9.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	12748	0
2	14446	0
3	17389	0
4	12713	0
5	12423	0
6	16446	0
7	16705	0
8	15910	0
9	15694	0
10	13069	0
11	17581	0
12	14051	0
13	17754	0

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM angelic-goods-456101-n9.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

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

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM angelic-goods-456101-n9.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID ▼	item_cnt ▼
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240
10	12356	1573

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

```
CREATE OR REPLACE TABLE angelic-goods-456101-n9.modulabs_project.user_rf AS
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM angelic-goods-456101-n9.modulabs_project.data
  GROUP BY CustomerID
),
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM angelic-goods-456101-n9.modulabs_project.data
  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 angelic-goods-456101-n9.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13436	1	76	1
3	13298	1	96	1
4	15520	1	314	1
5	14569	1	79	1
6	14204	1	72	2
7	15471	1	255	2
8	15195	1	1404	2
9	12442	1	181	3
10	15318	1	642	3

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity)) AS user_total
FROM angelic-goods-456101-n9.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.0
4	12349	1458.0
5	12350	294.0
6	12352	1265.0
7	12353	89.0
8	12354	1079.0
9	12355	459.0
10	12356	2487.0

- 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 FROM angelic-goods-456101-n9.modulab
s_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) AS user_average
FROM angelic-goods-456101-n9.modulabs_project.user_rf rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity)) AS user_total
  FROM angelic-goods-456101-n9.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13436	1	76	1	197.0	197.0
3	14569	1	79	1	227.0	227.0
4	13298	1	96	1	360.0	360.0
5	15520	1	314	1	343.0	343.0
6	15195	1	1404	2	3861.0	3861.0
7	15471	1	255	2	448.0	448.0
8	14204	1	72	2	151.0	151.0
9	12442	1	181	3	144.0	144.0
10	14578	1	240	3	169.0	169.0

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *  
FROM angelic-goods-456101-n9.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13436	1	76	1	197.0	197.0
3	14569	1	79	1	227.0	227.0
4	13298	1	96	1	360.0	360.0
5	15520	1	314	1	343.0	343.0
6	15195	1	1404	2	3861.0	3861.0
7	15471	1	255	2	448.0	448.0
8	14204	1	72	2	151.0	151.0
9	12442	1	181	3	144.0	144.0
10	14578	1	240	3	169.0	169.0

11-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

- 1) 고객 별로 구매한 상품들의 고유한 수를 계산하기
- 2) `user_rfm` 테이블과 결과를 합치기
- 3) `user_data` 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE `angelic-goods-456101-n9.modulabs_project.
user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_product_count
  FROM
    `angelic-goods-456101-n9.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
SELECT
  u.*,
  up.unique_product_count
FROM
  `angelic-goods-456101-n9.modulabs_project.user_rfm` AS u
LEFT JOIN
  unique_products AS up
ON
  u.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...
1	16738	1	3	297	4.0	4.0	1
2	13188	1	24	11	100.0	100.0	1
3	13307	1	4	120	15.0	15.0	1
4	18233	1	4	325	440.0	440.0	1
5	15316	1	100	326	165.0	165.0	1
6	17925	1	72	372	244.0	244.0	1
7	13829	1	-12	359	-102.0	-102.0	1
8	16138	1	-1	368	-8.0	-8.0	1
9	17347	1	216	86	229.0	229.0	1
10	15389	1	400	172	500.0	500.0	1

2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 평균 구매 소요 일수를 계산하고, 그 결과를 `user_data` 에 통합

```
CREATE OR REPLACE TABLE `angelic-goods-456101-n9.modulabs_project.
user_data` AS
WITH purchase_intervals AS (
  SELECT
    CustomerID,
    IFNULL(ROUND(AVG(interval_), 2), 0) AS average_interval
  FROM (
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY Custo
merID ORDER BY InvoiceDate), DAY) AS interval_
    FROM
      `angelic-goods-456101-n9.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT
  ud.*,
```

```

pi.average_interval
FROM
`angelic-goods-456101-n9.modulabs_project.user_data` AS ud
LEFT JOIN
purchase_intervals AS pi
ON
ud.CustomerID = pi.CustomerID;

```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...
1	17102	1	2	261	26.0	26.0	1	0.0
2	13188	1	24	11	100.0	100.0	1	0.0
3	17948	1	144	147	359.0	359.0	1	0.0
4	16144	1	16	246	175.0	175.0	1	0.0
5	16257	1	1	176	22.0	22.0	1	0.0
6	17307	1	-144	365	-153.0	-153.0	1	0.0
7	17752	1	192	359	81.0	81.0	1	0.0
8	13703	1	10	318	100.0	100.0	1	0.0
9	15562	1	39	351	135.0	135.0	1	0.0
10	17925	1	72	372	244.0	244.0	1	0.0

3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기
 - 1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
 - 2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
 - 취소 빈도와 취소 비율을 계산하고 그 결과를 **user_data**에 통합하기
(취소 비율은 소수점 두번째 자리)

```

CREATE OR REPLACE TABLE `angelic-goods-456101-n9.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 E

```

```

ND) AS cancel_frequency
FROM `angelic-goods-456101-n9.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
)

SELECT
  ud.*,
  ti.cancel_frequency,
  ROUND(ti.cancel_frequency / ti.total_transactions, 2) AS cancel_rate
FROM
  `angelic-goods-456101-n9.modulabs_project.user_data` AS ud
LEFT JOIN
  TransactionInfo AS ti
ON
  ud.CustomerID = ti.CustomerID;

```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...	cancel_frequ...	cancel_rate
1	15216	1	28	87	97.0	97.0	2	0.0	0	0.0
2	14457	1	88	310	114.0	114.0	4	0.0	0	0.0
3	18191	1	140	261	208.0	208.0	7	0.0	0	0.0
4	14863	1	132	29	267.0	267.0	11	0.0	0	0.0
5	17824	1	408	51	298.0	298.0	13	0.0	0	0.0
6	16509	1	64	296	364.0	364.0	13	0.0	0	0.0
7	13045	1	221	99	305.0	305.0	18	0.0	0	0.0
8	15699	1	113	17	310.0	310.0	18	0.0	0	0.0
9	15528	1	123	254	229.0	229.0	26	0.0	0	0.0
10	17551	1	142	359	290.0	290.0	39	0.0	0	0.0

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

```

SELECT *
FROM `angelic-goods-456101-n9.modulabs_project.user_data`;

```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_product_c...	average_interval
1	15216	1	28	87	97.0	97.0	2	0.0
2	14457	1	88	310	114.0	114.0	4	0.0
3	18191	1	140	261	208.0	208.0	7	0.0
4	14863	1	132	29	267.0	267.0	11	0.0
5	17824	1	408	51	298.0	298.0	13	0.0
6	16509	1	64	296	364.0	364.0	13	0.0
7	13045	1	221	99	305.0	305.0	18	0.0
8	15699	1	113	17	310.0	310.0	18	0.0
9	15528	1	123	254	229.0	229.0	26	0.0
10	17551	1	142	359	290.0	290.0	39	0.0

회고

[회고 내용을 작성해주세요]

Keep

- SQL을 활용해 단계별 데이터 전처리(결측치, 중복, 오류값)를 체계적으로 수행한 점이 좋았다.
- RFM 스코어(Recency, Frequency, Monetary)를 계산하여 고객 특성을 정량적으로 파악할 수 있었다.
- 추가 Feature(구매 다양성, 평균 구매 주기, 취소 경향성)를 반영함으로써 고객의 행동 패턴을 더 깊이 이해할 수 있었다.
- 최종적으로 `user_data` 테이블을 완성해 고객 세그멘테이션을 위한 기반 데이터셋을 확보했다.

Problem

- 데이터 전처리 과정에서 삭제/정제 기준을 설정하는 부분이 일관되지 않아 의사결정 시간이 오래 걸렸다.
- InvoiceNo, StockCode 등 일부 컬럼 처리 시 규칙 설정이 모호하여 불필요한 trial & error가 발생했다.
- RFM 외 추가 Feature의 해석과 활용 방안이 충분히 논의되지 않아, 비즈니스적으로 어떻게 연결할지 구체성이 부족했다.
- EDA와 전처리에서 시각화를 병행하지 않아, 인사이트 발견이 다소 제한적이었다.

Try

- 전처리 단계에서 기준을 사전에 명확히 정의(ex) 취소 데이터 처리, 가격=0 제거 기준) 하여 효율성을 높일것..!

- RFM 분석 이후 K-means, Hierarchical Clustering 등 실제 세그멘테이션 알고리즘을 적용해 군집 결과를 도출해보기 (이미 했음)
- 추가 Feature를 바탕으로 각 고객 세그먼트의 특성을 구체적으로 해석하고, 마케팅 전략/추천 시스템 등에 연결해본다.
- 분석 과정에서 데이터 시각화를 적극 활용하여 전처리 및 Feature 중요성을 직관적으로 확인할 수 있도록 개선한다.

너무 오랜만에 봐서 기억이 잘 안난다..!!