# 고객을 세그먼테이션하자 [프로젝트]

## 11-2. 데이터 불러오기

#### 데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

SELECT \* FROM zeta-ascent-425501-j3.modulabs.data LIMIT 10

행 //	InvoiceNo ▼	StockCode ▼	Description ▼	Quantity 🕶	InvoiceDate ▼	UnitPrice ▼	CustomerID ▼	Country ▼
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	nuli	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	nuli	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	nul	United Kingdom
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	nul	United Kingdom
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	nul	United Kingdom
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	nul	United Kingdom
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	0.0	nuli	United Kingdom
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	nuli	United Kingdom
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	0.0	nul	United Kingdom
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	0.0	nul	United Kingdom

• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*) FROM zeta-ascent-425501-j3.modulabs.data
```

#### 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 zeta-ascent-425501-j3.modulabs.data



## 11-4. 데이터 전처리 방법(1): 결측치 제거

#### 컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계산
  - $\circ$  각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

SELECT 'Description' AS column\_name, COUNT(\*) AS null\_count FROM zeta-ascent-425501-j3.modulabs.data WHERE Description IS NULL

```
UNION ALL

SELECT 'CustomerID' AS column_name, COUNT(*) AS null_count
FROM zeta-ascent-425501-j3.modulabs.data
WHERE CustomerID IS NULL;
```



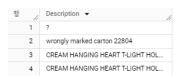
#### 결측치 처리 전략

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

```
SELECT Description

FROM zeta-ascent-425501-j3.modulabs.data

WHERE StockCode = '85123A'
```



#### 결측치 처리

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

```
DELETE FROM zeta-ascent-425501-j3.modulabs.data
WHERE CustomerID IS NULL;

DELETE FROM zeta-ascent-425501-j3.modulabs.data
WHERE Description IS NULL
```

● 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 SUM(duplicate_count) AS total_duplicates

FROM (

SELECT

COUNT(*) AS duplicate_count

FROM

zeta-ascent-425501-j3.modulabs.data

GROUP BY

InVoiceNo,
StockCode,
Description,
```



#### 중복값 처리

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

```
CREATE OR REPLACE TABLE `modulabs.data` AS
SELECT DISTINCT *
FROM zeta-ascent-425501-j3.modulabs.data;

SELECT COUNT(*)
FROM zeta-ascent-425501-j3.modulabs.data;
```



## 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 zeta-ascent-425501-j3.modulabs.data;



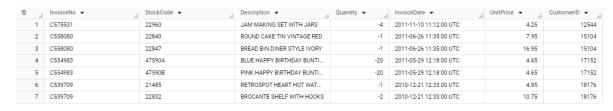
• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM zeta-ascent-425501-j3.modulabs.data
LIMIT 100;
```



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

SELECT \*
FROM zeta-ascent-425501-j3.modulabs.data
WHERE InVoiceNO LIKE "C%"
LIMIT 100;



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

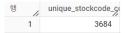
SELECT ROUND(SUM(CASE WHEN InVoiceNO LIKE "C%" THEN 1 ELSE 0 END)/ COUNT(\*) \* 100, 1) AS canceled\_ra FROM zeta-ascent-425501-j3.modulabs.data;



#### StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode) AS unique\_stockcode\_count FROM zeta-ascent-425501-j3.modulabs.data;



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

SELECT StockCode, COUNT(\*) AS sell\_cnt FROM zeta-ascent-425501-j3.modulabs.data 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 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 zeta-ascent-425501-j3.modulabs.data
)
WHERE number_count <= 1</pre>
```

핸.	StockCode ▼	number_count ▼
- //	//	//
1	POST	0
2	M	0
3	PADS	0
4	D	0
5	BANK CHARGES	0
6	DOT	0
7	CRUK	0
8	C2	1

- 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 DISTINCT ROUND(COUNT(number_count) * 100 / 401604, 2) AS percentage
FROM (
    SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM zeta-ascent-425501-j3.modulabs.data
)
WHERE number_count <= 1</pre>
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM zeta-ascent-425501-j3.modulabs.data

WHERE StockCode IN (

SELECT DISTINCT StockCode

FROM (

SELECT StockCode,

LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count

FROM zeta-ascent-425501-j3.modulabs.data
) AS subquery
```

```
WHERE number_count <= 1
);</pre>
```

① 이 문으로 data의 행 1,915개가 삭제되었습니다.

### Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS frequency
FROM zeta-ascent-425501-j3.modulabs.data
GROUP BY Description
ORDER BY frequency DESC
LIMIT 30;
```

핼 //	Description ▼	frequency ▼
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
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000
14	HEART OF WICKER SMALL	990
15	NATURAL SLATE HEART CHAL	989
16	JAM MAKING SET WITH JARS	966
17	LUNCH BAG PINK POLKADOT	961
18	LUNCH BAG SUKI DESIGN	932
19	ALARM CLOCK BAKELIKE RED	917
20	REX CASH+CARRY JUMBO SH	900

• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM zeta-ascent-425501-j3.modulabs.data
WHERE LOWER(Description) LIKE '%next day carriage%'
OR LOWER(Description) LIKE '%high resolution image%';
```

● 이 문으로 data의 행 83개가 삭제되었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE zeta-ascent-425501-j3.modulabs.data1 AS
SELECT
InvoiceNo,
StockCode,
Quantity,
InvoiceDate,
UnitPrice,
CustomerID,
```

고객을 세그먼테이션하자 [프로젝트]

6

Country,
UPPER(Description) AS Description
FROM zeta-ascent-425501-j3.modulabs.data;

● 이 문으로 이름이 data1인 새 테이블이 생성되었습니다.

#### 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 zeta-ascent-425501-j3.modulabs.data1



• 단가가 0원인 거래의 개수, 구매 수량( quantity )의 최솟값, 최댓값, 평균 구하기

SELECT MIN(Quantity) AS  $min_quantity$ , Max(Quantity) AS  $max_quantity$ , AVG(Quantity) AS  $avg_quantity$  FROM zeta-ascent-425501-j3.modulabs.data1 WHERE UnitPrice = 0



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE zeta-ascent-425501-j3.modulabs.data2 AS SELECT \*
FROM zeta-ascent-425501-j3.modulabs.data1
WHERE UnitPrice != 0

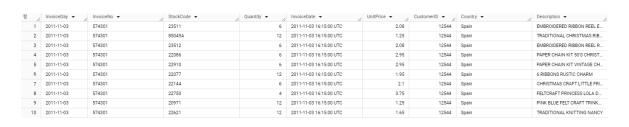
● 이 문으로 이름이 data2인 새 테이블이 생성되었습니다.

### 11-7. RFM 스코어

#### Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InVoiceDate) AS InvoiceDay, \*
FROM zeta-ascent-425501-j3.modulabs.data2;



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT MAX(InvoiceDate) AS latest_purchase_date
FROM zeta-ascent-425501-j3.modulabs.data2;
```



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
MAX(InVoiceDate) AS InvoiceDay
FROM zeta-ascent-425501-j3.modulabs.data2
GROUP BY CustomerID;
```

행 //	CustomerID ▼	InvoiceDay ▼
1	12544	2011-11-10 11:12:00 UTC
2	13568	2011-06-19 14:42:00 UTC
3	13824	2011-11-07 12:41:00 UTC
4	14080	2011-11-07 11:09:00 UTC
5	14336	2011-11-23 11:40:00 UTC
6	14592	2011-11-04 16:35:00 UTC
7	15104	2011-06-26 11:35:00 UTC
8	15360	2011-10-31 09:35:00 UTC
9	15872	2011-11-25 11:55:00 UTC
10	16128	2011-11-22 11:14:00 UTC

• 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
);
```

[결과 이미지를 넣어주세요]

• 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
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

#### Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
# [[YOUR QUERY]] AS purchase_cnt
```

```
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
# [[YOUR QUERY]] AS item_cnt
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];

[결과 이미지를 넣어주세요]
```

• 전체 거래 건수 계산와 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 (
 # [[YOUR QUERY]]
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
 # [[YOUR QUERY]]
-- 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;
```

[결과 이미지를 넣어주세요]

#### Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
# [[YOUR QUERY]] AS user_total
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

- 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,
    # [[YOUR QUERY]] AS user_average

FROM project_name.modulabs_project.user_rf rf

LEFT JOIN (
    -- 고객 별 총 지출액
    SELECT
    # [[YOUR QUERY]]
) ut

ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

#### RFM 통합 테이블 출력하기

• 최종 user\_rfm 테이블을 출력하기

```
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

## 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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
     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;
```

[결과 이미지를 넣어주세요]

#### 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)

[결과 이미지를 넣어주세요]

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

# [[YOUR QUERY]];

[결과 이미지를 넣어주세요]