

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

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

```
SELECT *  
FROM striped-rhino-466601-k6.modulabs_project.data  
LIMIT 10
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

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

```
SELECT COUNT(*)  
FROM striped-rhino-466601-k6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	f0_
1	541909

데이터 수 세기

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

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

```
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 striped-rhino-466601-k6.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 ALL**을 통해 합치기

```
# [[YOUR QUERY]]

SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM striped-rhino-466601-k6.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 striped-rhino-466601-k6.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 striped-rhino-466601-k6.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 striped-rhino-466601-k6.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 striped-rhino-466601-k6.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 striped-rhino-466601-k6.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 striped-rhino-466601-k6.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 striped-rhino-466601-k6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	column_name	missing_percentage
1	CustomerID	24.93
2	Country	0.0
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

[[YOUR QUERY]]

```
SELECT DISTINCT Description
FROM striped-rhino-466601-k6.modulabs_project.data
WHERE StockCode = '85123A'
ORDER BY Description
```

[결과 이미지를 넣어주세요]

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

결측치 처리


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

[[YOUR QUERY]]

```
DELETE FROM striped-rhino-466601-k6.modulabs_project.data
WHERE Description IS NULL
OR CustomerID IS NULL
```

[결과 이미지를 넣어주세요]

작업 정보
결과
실행 세부정보
실행 그래프

 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

[[YOUR QUERY]]

```
SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, COUNT(*) AS cnt
FROM striped-rhino-466601-k6.modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING cnt > 1
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	cnt
1	571034	23239	SET OF 4 ...	6	2011-10-13...	4.15	12359	Cyprus	2
2	571034	23245	SET OF 3 R...	4	2011-10-13...	4.95	12359	Cyprus	2
3	571034	23494	VINTAGE ...	3	2011-10-13...	5.95	12359	Cyprus	2
4	538826	22749	FELTCRAF...	1	2010-12-14...	3.75	12370	Cyprus	2
5	577228	23048	SET OF 10 ...	1	2011-11-18...	4.15	12391	Cyprus	3
6	577228	22435	SET OF 9 ...	1	2011-11-18...	1.25	12391	Cyprus	2
7	577228	22270	HAPPY EA...	1	2011-11-18...	3.75	12391	Cyprus	2
8	577228	84580	MOUSE TO...	1	2011-11-18...	3.75	12391	Cyprus	2
9	577228	23156	SET OF 5 ...	1	2011-11-18...	2.08	12391	Cyprus	2
10	577228	22144	CHRISTM...	1	2011-11-18...	2.1	12391	Cyprus	2
11	539419	48138	DOORMAT...	10	2010-12-17...	6.75	12431	Austra...	2

페이지당 결과 수: 50 1 - 5 (전체 4837행)

중복값 처리

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

[[YOUR QUERY]]

```
-- 중복 제거하여 data2라는 이름의 테이블 생성
CREATE OR REPLACE TABLE striped-rhino-466601-k6.modulabs_project.data2 AS
SELECT DISTINCT *
FROM striped-rhino-466601-k6.modulabs_project.data
```

```
-- data2 테이블의 행 갯수 조회
SELECT COUNT(*)
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

쿼리 결과	결과 저장	다음에서 열기
작업 정보	결과	실행 세부정보
이 문으로 이름이 data2인 새 테이블이 생성되었습니다.		
테이블로 이동		

쿼리 결과	
작업 정보 결과	
행	f0_
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

[[YOUR QUERY]]

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

작업 정보 결과	
행	f0_
1	22190

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

[[YOUR QUERY]]

```
SELECT InvoiceNo
FROM striped-rhino-466601-k6.modulabs_project.data2
LIMIT 100
```

[결과 이미지를 넣어주세요]

작업 정보 결과		차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
1	541431				
2	C541433				
3	537626				
4	537626				
5	537626				
6	537626				
7	537626				

페이지당 결과 수: 50 ▼ 1 ~ 50 전체 100행

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

[[YOUR QUERY]]

```
SELECT *
FROM striped-rhino-466601-k6.modulabs_project.data2
```

WHERE InvoiceNo LIKE 'C%'
LIMIT 100

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM ...	-74215	2011-01-18 1...	1.04	12346	United Kin...
2	C545329	M	Manual	-1	2011-03-01 1...	280.05	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 1...	183.75	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 1...	376.5	12352	Norway
5	C547388	84050	PINK HEA...	-12	2011-03-22 1...	1.65	12352	Norway
6	C547388	37448	CERAMIC ...	-12	2011-03-22 1...	1.49	12352	Norway
7	C547388	22645	CERAMIC ...	-12	2011-03-22 1...	1.45	12352	Norway

페이지당 결과 수: 50 1 ~ 50 전체 100행

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

[[YOUR QUERY]]

```
SELECT ROUND( SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(InvoiceNo) *100, 1 ) AS InvoiceNo_cancelled
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

행	InvoiceNo_cancelled
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

[[YOUR QUERY]]

```
SELECT COUNT(DISTINCT StockCode) AS StockCode_unique_count
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

행	StockCode_unique_count
1	3684

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

[[YOUR QUERY]]

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM striped-rhino-466601-k6.modulabs_project.data2
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

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

[[YOUR QUERY]]

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM striped-rhino-466601-k6.modulabs_project.data2
)
WHERE number_count=0 OR number_count=1
```

[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)

[[YOUR QUERY]]

```
-- number_count가 0 또는 1인 StockCode 조회
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM striped-rhino-466601-k6.modulabs_project.data2
)
WHERE number_count = 0 OR number_count = 1

-- 조회 결과 나오는 StockCode가 전체 데이터에서 차지하는 비율
SELECT ROUND( SUM(CASE WHEN StockCode='POST'
```

```

OR StockCode='D'
OR StockCode='C2'
OR StockCode='M'
OR StockCode='BANK CHARGES'
OR StockCode='PADS'
OR StockCode='DOT'
OR StockCode='CRUK' THEN 1 ELSE 0 END) / COUNT(StockCode) *100, 2 ) AS StockCode_outlier_percentage
FROM striped-rhino-466601-k6.modulabs_project.data2

```

[결과 이미지를 넣어주세요]

작업 정보		결과	차트	JS
행	StockCode_outlier_percentage ▾			
1	0.48			

- 제품과 관련되지 않은 거래 기록을 제거하기

```

# [[YOUR QUERY]]

DELETE FROM striped-rhino-466601-k6.modulabs_project.data2
WHERE StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK')

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보
결과
실행 세부정보
실행 그래프

i 이 문으로 data2의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```

# [[YOUR QUERY]]

SELECT Description, COUNT(Description) AS description_cnt
FROM striped-rhino-466601-k6.modulabs_project.data2
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30

```

[결과 이미지를 넣어주세요]

행	Description	description_cnt
1	WHITE HA...	2058
2	REGENCY ...	1894
3	JUMBO BA...	1659
4	PARTY BU...	1409
5	ASSORTED...	1405
6	LUNCH BA...	1345
7	SET OF 3 C...	1224
8	LUNCH BA...	1099
9	PACK OF 7...	1062
10	SPOTTY B...	1026

페이지당 결과 수: 50 1 - 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

[[YOUR QUERY]]

```
DELETE FROM striped-rhino-466601-k6.modulabs_project.data2
WHERE Description IN ('Next Day Carriage', 'High Resolution Image')
```

[결과 이미지를 넣어주세요]

작업 정보
결과
실행 세부정보
실행 그래프

i 이 문으로 data2의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

[[YOUR QUERY]]

```
CREATE OR REPLACE TABLE striped-rhino-466601-k6.modulabs_project.data2 AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

작업 정보
결과
실행 세부정보
실행 그래프

i 이 문으로 이름이 data2인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- UnitPrice 의 최솟값, 최댓값, 평균을 구하기

[[YOUR QUERY]]

```
SELECT MIN(UnitPrice) AS min_price
      , MAX(UnitPrice) AS max_price
      , AVG(UnitPrice) AS avg_price
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

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

[[YOUR QUERY]]

```
SELECT COUNT(Quantity) AS cnt_quantity
, MIN(Quantity) AS min_quantity
, MAX(Quantity) AS max_quantity
, AVG(Quantity) AS avg_quantity
FROM striped-rhino-466601-k6.modulabs_project.data2
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

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

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

[[YOUR QUERY]]

```
CREATE OR REPLACE TABLE striped-rhino-466601-k6.modulabs_project.data2 AS
SELECT *
FROM striped-rhino-466601-k6.modulabs_project.data2
WHERE UnitPrice != 0
```

[결과 이미지를 넣어주세요]

작업 정보
결과
실행 세부정보
실행 그래프

i 이 문으로 이름이 data2인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

[[YOUR QUERY]]

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP S
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP S
3	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	GREEN DRAWER KNOB AC
4	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB ACR
5	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE
6	2010-12-07	537626	20782	6	2010-12-07 14:57:00 UTC	5.49	12347	Iceland	CAMOUFLAGE EAR MUFF
7	2010-12-07	537626	84969	6	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	BOX OF 6 ASSORTED COL
8	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	RED DRAWER KNOB ACRY
9	2010-12-07	537626	22728	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE
10	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	CLEAR DRAWER KNOB AC

페이지당 결과 수: 50 1 - 50 (전체 399573행)

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

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

```
SELECT MAX(InvoiceDate) AS most_recent_date
FROM striped-rhino-466601-k6.modulabs_project.data2
```

[결과 이미지를 넣어주세요]

행	most_recent_date
1	2011-12-09

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

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

```
SELECT CustomerID
, MAX(InvoiceDate) AS most_recent_date
FROM striped-rhino-466601-k6.modulabs_project.data2
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

행	CustomerID	most_recent_date
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

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(InvoiceDate) AS InvoiceDay
FROM striped-rhino-466601-k6.modulabs_project.data2
GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	12395	15
2	12509	284
3	12901	8
4	13145	39
5	13209	30
6	13218	127
7	14434	21
8	14481	164
9	14514	61
10	14772	112

페이지당 결과 수: 50 1 - 50 (전체 4362행)

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

[[YOUR QUERY]]

```
CREATE OR REPLACE TABLE striped-rhino-466601-k6.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 striped-rhino-466601-k6.modulabs_project.data2
  GROUP BY CustomerID
)
```


[결과 이미지를 넣어주세요]

작업 정보

결과

실행 세부정보

실행 그래프

 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

[[YOUR QUERY]]

```
SELECT CustomerID
  , COUNT(InvoiceNo) AS purchase_cnt
FROM striped-rhino-466601-k6.modulabs_project.data2
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84
7	12353	4
8	12354	58
9	12355	13
10	12356	58

페이지당 결과 수: 200 1 - 200 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

[[YOUR QUERY]]

```
SELECT CustomerID
      , SUM(Quantity) AS item_cnt
FROM striped-rhino-466601-k6.modulabs_project.data2
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

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

[[YOUR QUERY]]

```
CREATE OR REPLACE TABLE striped-rhino-466601-k6.modulabs_project.user_rf AS
```

```
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT CustomerID, COUNT(InvoiceNo) AS purchase_cnt
  FROM striped-rhino-466601-k6.modulabs_project.data2
  GROUP BY CustomerID
),
```

```
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT CustomerID, SUM(Quantity) AS item_cnt
  FROM striped-rhino-466601-k6.modulabs_project.data2
  GROUP BY CustomerID
)
```

```
-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.receency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN striped-rhino-466601-k6.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency
1	12346	2	0	325
2	12347	182	2458	2
3	12348	27	2332	75
4	12349	72	630	18
5	12350	16	196	310
6	12352	84	463	36
7	12353	4	20	204
8	12354	58	530	232
9	12355	13	240	214
10	12356	58	1573	22

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

[[YOUR QUERY]]

```
SELECT CustomerID
, ROUND(SUM(UnitPrice), 1) AS user_total
FROM striped-rhino-466601-k6.modulabs_project.data2
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	12346	2.1
2	12347	481.2
3	12348	18.7
4	12349	305.1
5	12350	25.3
6	12352	330.5
7	12353	24.3
8	12354	261.2
9	12355	54.7
10	12356	170.9

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_cnt로 나누어서 3) user_rfm 테이블로 저장하기

[[YOUR QUERY]]

```
CREATE OR REPLACE TABLE striped-rhino-466601-k6.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 striped-rhino-466601-k6.modulabs_project.user_rf rf
LEFT JOIN (
SELECT CustomerID, ROUND(SUM(UnitPrice), 1) AS user_total
FROM striped-rhino-466601-k6.modulabs_project.data2
GROUP BY CustomerID
```

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

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	15311	2478	37673	0	6245.1	2.5
2	16705	284	5458	0	1284.3	4.5
3	13426	158	2225	0	300.2	1.9
4	12423	118	1312	0	245.0	2.1
5	14422	222	2906	0	623.9	2.8
6	12662	222	2023	0	561.5	2.5
7	17364	409	2671	0	1114.9	2.7
8	13069	469	5454	0	809.1	1.7
9	15804	273	2513	0	1121.8	4.1
10	15910	266	1013	0	482.3	1.8

페이지당 결과 수: 50 1 - 50 (전체 4362행)

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

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

```
SELECT *
FROM striped-rhino-466601-k6.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12875	2	2019	143	0.3	0.1
2	15118	1	1440	134	0.2	0.2
3	15744	3	120	77	0.9	0.3
4	12891	3	950	185	1.1	0.4
5	17752	1	192	359	0.4	0.4
6	13027	26	17280	113	10.4	0.4
7	14124	5	1618	84	1.8	0.4
8	13678	16	924	151	6.3	0.4
9	16377	5	1500	267	2.7	0.5
10	13848	5	3700	92	2.3	0.5

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

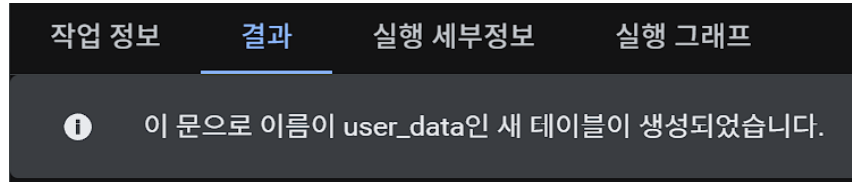
```
CREATE OR REPLACE TABLE striped-rhino-466601-k6.modulabs_project.user_data AS
WITH unique_products AS (
SELECT
  CustomerID,
```

```

COUNT(DISTINCT StockCode) AS unique_products
FROM striped-rhino-466601-k6.modulabs_project.data2
GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM striped-rhino-466601-k6.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

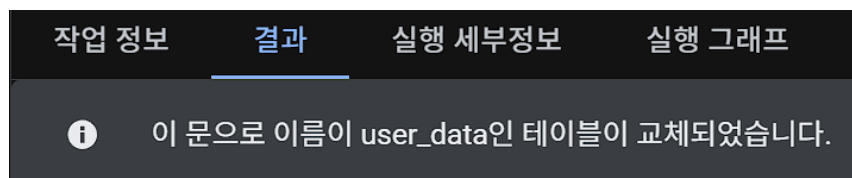
```

CREATE OR REPLACE TABLE striped-rhino-466601-k6.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
      striped-rhino-466601-k6.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM striped-rhino-466601-k6.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
 - 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 striped-rhino-466601-k6.modulabs_project.user_data2 AS

WITH TransactionInfo AS (
  SELECT CustomerID
    , InvoiceNo
    , CASE WHEN InvoiceNo Like 'C%' THEN 1 ELSE 0
      END AS is_cancelled
  FROM striped-rhino-466601-k6.modulabs_project.data3
),

CustomerSummary AS (
  SELECT CustomerID
    , COUNT(*) AS total_transaction
    , SUM(is_cancelled) AS cancel_frequency
    , ROUND(SAFE_CAST(SUM(is_cancelled) AS FLOAT64) / COUNT(*), 2) AS cancel_rate
  FROM TransactionInfo
  GROUP BY CustomerID
)

SELECT u.*
  , cs.total_transaction
  , cs.cancel_frequency
  , cs.cancel_rate
FROM CustomerSummary AS cs
JOIN striped-rhino-466601-k6.modulabs_project.user_data AS u
ON cs.CustomerID = u.CustomerID
```

[결과 이미지를 넣어주세요]

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

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

```
SELECT *
FROM striped-rhino-466601-k6.modulabs_project.user_data2
```

[결과 이미지를 넣어주세요]

작업 정보		결과	차트	JSON	실행 세부정보		실행 그래프				
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transaction	cancel_frequency	cancel_rate
1	15316	1	100	326	1.6	1.6	1	0.0	1	0	0.0
2	15389	1	400	172	1.3	1.3	1	0.0	1	0	0.0
3	16144	1	16	246	10.9	10.9	1	0.0	1	0	0.0
4	15753	1	144	304	0.6	0.6	1	0.0	1	0	0.0
5	17347	1	216	86	1.1	1.1	1	0.0	1	0	0.0
6	16990	1	100	218	1.8	1.8	1	0.0	1	0	0.0
7	18133	1	1350	212	0.7	0.7	1	1.0	1	0	0.0
페이지당 결과 수: 50 1 ~ 50 (전체 4362명) < > >>											

회고

Pandas로도 해봐야겠다는 생각이 듭니다...