

# 고객을 세그멘테이션하자! [프로젝트] - 이택원

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM `smiling-mark-468901-i5.modulabs_project.data`  
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

	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

```
SELECT COUNT(*) AS total_rows  
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	total_rows				
1	541909				

### 데이터 수 세기

- SELECT  
COUNT(InvoiceNo) AS cnt\_InvoiceNo,  
COUNT(StockCode) AS cnt\_StockCode,  
COUNT(Description) AS cnt\_Description,  
COUNT(Quantity) AS cnt\_Quantity,  
COUNT(InvoiceDate) AS cnt\_InvoiceDate,  
COUNT(UnitPrice) AS cnt\_UnitPrice,  
COUNT(CustomerID) AS cnt\_CustomerID,  
COUNT(Country) AS cnt\_Country  
FROM `smiling-mark-468901-i5.modulabs\_project.data`;

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

	cnt_InvoiceNo	cnt_StockCode	cnt_Description	cnt_Quantity	cnt_InvoiceDate	cnt_UnitPrice	cnt_CustomerID	cnt_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

```
-- 각 컬럼의 누락값 비율(%) 계산 (소수점 2자리)
WITH base AS (
  SELECT *
  FROM `smiling-mark-468901-i5.modulabs_project.data`
)
SELECT 'InvoiceNo' AS column_name,
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2) AS missing_percentage
FROM base
UNION ALL
SELECT 'StockCode',
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'Description',
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'Quantity',
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'InvoiceDate',
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'UnitPrice',
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'CustomerID',
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
UNION ALL
SELECT 'Country',
       ROUND(SAFE_DIVIDE(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END), COUNT(1)) * 100, 2)
FROM base
ORDER BY missing_percentage DESC;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	column_name ▼	missing_percenta...			
1	CustomerID	24.93			
2	Description	0.27			
3	InvoiceNo	0.0			
4	InvoiceDate	0.0			
5	StockCode	0.0			
6	Country	0.0			
7	UnitPrice	0.0			
8	Quantity	0.0			

## 결측치 처리 전략

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

```
SELECT DISTINCT
Description
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
	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 `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NULL
OR Description IS NULL;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
	<p><b>i</b> 이 문으로 data의 행 135,080개가 삭제되었습니다.</p>		

## 11-5. 데이터 전처리(2): 중복값 처리

### 중복값 확인

- 중복된 행의 수를 세어보기

- 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

```

---- 1) 중복 그룹과 중복 행 수 집계
WITH grouped AS (
  SELECT
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country,
    COUNT(*) AS cnt
  FROM `smiling-mark-468901-i5.modulabs_project.data`
  GROUP BY 1,2,3,4,5,6,7,8
)
SELECT
  -- 중복 그룹 개수(동일 조합이 2번 이상 등장한 조합의 개수)
  SUM(CASE WHEN cnt > 1 THEN 1 ELSE 0 END) AS duplicate_groups,
  -- 중복이 포함된 전체 행 수(해당 그룹의 행 수 합계)
  SUM(CASE WHEN cnt > 1 THEN cnt ELSE 0 END) AS rows_in_duplicate_groups,
  -- '중복'으로 간주되는 초과 행 수(각 그룹에서 1개를 제외한 나머지)
  SUM(CASE WHEN cnt > 1 THEN cnt - 1 ELSE 0 END) AS excess_duplicate_rows
FROM grouped;

```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	duplicate_groups	rows_in_duplicate...	excess_duplicate...		
1	4837	10062	5225		

## 중복값 처리

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

```

--중복값 제거 쿼리

CREATE OR REPLACE TABLE `smiling-mark-468901-i5.modulabs_project.data` AS
SELECT DISTINCT *
FROM `smiling-mark-468901-i5.modulabs_project.data`;

--중복 제거 후 행 개수 확인 쿼리

SELECT COUNT(*) AS total_rows_after_dedup
FROM `smiling-mark-468901-i5.modulabs_project.data`;

```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p><b>i</b> 이 문으로 이름이 data인 테이블이 교체되었습니다.</p>			

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	total_rows_after_...				
1	401604				

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count  
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	unique_invoice_c...				
1	22190				

- 고유한 InvoiceNo를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo  
FROM `smiling-mark-468901-i5.modulabs_project.data`  
LIMIT 100;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과 결과 저장

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
InvoiceNo					
1	541431				
2	541433				
3	537626				
4	542237				
5	549222				
6	556201				
7	562032				
8	572511				
9	581180				
10	539318				
11	541998				
12	548955				

페이지당 결과 수: 50 1 ~ 50 (전체 100행)

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

```
SELECT *  
FROM `smiling-mark-468901-i5.modulabs_project.data`  
WHERE InvoiceNo LIKE 'C%'  
LIMIT 100;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	CS41433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	CS45329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3	CS45329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4	CS45330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	CS47388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
6	CS47388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
7	CS47388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
8	CS47388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
9	CS47388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
10	CS47388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
11	CS47388	22845	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway

페이지당 결과 수: 50 1 - 50 (전체 100행)

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

```
SELECT
  ROUND(
    COUNTIF(InvoiceNo LIKE 'C%') / COUNT(*) * 100,
    1
  ) AS cancelled_ratio_percent
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	cancelled_ratio_p...
1	2.2

## StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	unique_stockcod...
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 `smiling-mark-468901-i5.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

## ← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
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 `smiling-mark-468901-i5.modulabs_project.data`
)
WHERE number_count IN (0, 1)
ORDER BY number_count, StockCode;
```

[결과 이미지를 넣어주세요]

## 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	StockCode	number_count			
1	BANK CHARGES	0			
2	CRUK	0			
3	D	0			
4	DOT	0			
5	M	0			
6	PADS	0			
7	POST	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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
WITH marked AS (
  SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `smiling-mark-468901-i5.modulabs_project.data`
)
SELECT
```

```
ROUND(COUNTIF(number_count IN (0, 1)) / COUNT(*) * 100, 2) AS ratio_percent
FROM marked;
```

[결과 이미지를 넣어주세요]

#### 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	ratio_percent				
1	0.48				

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `smiling-mark-468901-i5.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 `smiling-mark-468901-i5.modulabs_project.data`
  )
  WHERE number_count IN (0, 1)
);
```

[결과 이미지를 넣어주세요]

#### 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p><b>i</b> 이 문으로 data의 행 1,915개가 삭제되었습니다.</p>			

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT
  Description,
  COUNT(*) AS description_cnt
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE Description IS NOT NULL
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]



쿼리 결과

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	Description	description_cnt
1	WHITE HANGING HEART TULG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1459
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORNL...	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
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 SHO...	900

페이지당 결과 수: 50 1 - 50 (전체 50행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 data의 행 83개가 삭제되었습니다.
-----------------------------

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `smiling-mark-468901-i5.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

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,
  ROUND(AVG(UnitPrice), 2) AS avg_price
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

## 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
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(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  ROUND(AVG(Quantity), 2) AS avg_quantity
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

## 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
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 `smiling-mark-468901-i5.modulabs_project.data` AS
SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE UnitPrice <> 0;
```

[결과 이미지를 넣어주세요]

## 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p><b>i</b> 이 문으로 이름이 data인 테이블이 교체되었습니다.</p>			

## 11-7. RFM 스코어

### Recency

- InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프				
일	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	33166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TI
2	2011-01-18	541433	33166	74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TI
3	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB
4	2010-12-07	537626	21731	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	RED TOASTTOOL LEE
5	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	SET OF 2 TINS VINTA
6	2010-12-07	537626	20782	6	2010-12-07 14:57:00 UTC	5.49	12347	Iceland	CAMOUFLAGE EAR M
7	2010-12-07	537626	22772	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	PINK DRAWER KNOB
8	2010-12-07	537626	22375	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	AIRLINE BAG VINTAG
9	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKE
10	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	RED DRAWER KNOB /
11	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland	MINI PAINT SET VINT
12	2010-12-07	537626	22716	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	LARGE HEART MESH
13	2010-12-07	537626	71477	12	2010-12-07 14:57:00 UTC	3.25	12347	Iceland	COLOUR GLASS STA
14	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	BLUE 3 PIECE POLKA
15	2010-12-07	537626	85232D	3	2010-12-07 14:57:00 UTC	4.95	12347	Iceland	SET/3 DECUPPAGE S
16	2010-12-07	537626	85147B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLACK GRAND BARD
17	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	GREEN DRAWER KNO
18	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	PINK 3 PIECE POLKA
19	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC	2.95	12347	Iceland	3D DOG PICTURE PL
20	2010-12-07	537626	21064	6	2010-12-07 14:57:00 UTC	5.95	12347	Iceland	BOOM BOX SPEAKER

페이지당 결과 수: 501 - 50 (전체 399573행) | < > |

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  MAX(DATE(InvoiceDate)) OVER() AS most_recent_date, -- 윈도우 함수
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `smiling-mark-468901-i5.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

#	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-12-09	2011-04-21	550911	23810	1	2011-04-21 13:11:00 UTC	16.95	12354	Spain
2	2011-12-09	2011-02-07	543370	21054	1	2011-02-07 14:51:00 UTC	8.95	12359	Cyprus
3	2011-12-09	2011-03-24	547684	23807	3	2011-03-24 14:46:00 UTC	16.95	12408	Belgium
4	2011-12-09	2011-03-03	545475	22715	144	2011-03-03 10:09:00 UTC	0.36	12415	Australia
5	2011-12-09	2011-06-15	556917	23233	200	2011-06-15 13:37:00 UTC	0.34	12415	Australia
6	2011-12-09	2011-06-15	556917	21508	72	2011-06-15 13:37:00 UTC	0.36	12415	Australia
7	2011-12-09	2011-06-15	556917	21929	100	2011-06-15 13:37:00 UTC	1.79	12415	Australia
8	2011-12-09	2011-10-05	568650	23926	180	2011-10-05 12:44:00 UTC	5.55	12415	Australia
9	2011-12-09	2011-06-20	557427	23110	6	2011-06-20 12:14:00 UTC	5.75	12429	Denmark

페이지당 결과 수: 50

1 - 50 (전체 399573행)

<

>

>>

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY InvoiceDay DESC;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
일	CustomerID	InvoiceDay			
1	12423	2011-12-09			
2	12433	2011-12-09			
3	12518	2011-12-09			
4	12526	2011-12-09			
5	12662	2011-12-09			
6	12680	2011-12-09			
7	12713	2011-12-09			
8	12748	2011-12-09			
9	12985	2011-12-09			
10	13069	2011-12-09			
11	13113	2011-12-09			
12	13426	2011-12-09			
13	13777	2011-12-09			
14	14051	2011-12-09			
15	14397	2011-12-09			
16	14422	2011-12-09			
17	14441	2011-12-09			
18	14446	2011-12-09			
19	15311	2011-12-09			
20	15344	2011-12-09			

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(DATE(InvoiceDate)) AS InvoiceDay
  FROM smiling-mark-468901-i5.modulabs_project.data
  GROUP BY CustomerID
);

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
명	CustomerID	recency			
1	12408	32			
2	12535	91			
3	12876	57			
4	13093	267			
5	13142	19			
6	13253	156			
7	13341	260			
8	13533	182			
9	13758	11			
10	13771	64			
11	13925	36			
12	14145	46			
13	14248	318			
14	14280	196			
15	14633	266			
16	14880	77			
17	14923	50			
18	15005	15			
19	15060	8			
20	15088	21			

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 `smiling-mark-468901-i5.modulabs_project.user_r` AS
WITH base AS (
  SELECT *
  FROM `smiling-mark-468901-i5.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  AND Description IS NOT NULL
  AND UnitPrice <> 0
  AND (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) NOT IN (0, 1)
),
last_purchase AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM base
  GROUP BY CustomerID
),
most_recent AS (
  SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
  FROM base
)
SELECT
  lp.CustomerID,
  lp.InvoiceDay AS last_purchase_date,
  mr.most_recent_date,
  DATE_DIFF(mr.most_recent_date, lp.InvoiceDay, DAY) AS recency_days
FROM last_purchase lp
CROSS JOIN most_recent mr;

```

[결과 이미지를 넣어주세요]

쿼리 결과			
작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 user_r인 새 데이터볼이 생성되었습니다.			

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY purchase_cnt DESC;
```

[결과 이미지를 넣어주세요]

쿼리 결과			
작업 정보	결과	시각화	JSON
실행 세부정보	실행 그래프		
명	CustomerID	purchase_cnt	
1	14911	242	
2	12748	217	
3	17841	169	
4	14606	125	
5	13089	118	
6	15311	118	
7	12971	88	
8	13408	75	
9	14646	73	
10	16029	66	
11	14156	64	
12	13798	63	
13	16422	60	
14	18102	60	
15	13694	57	
16	14527	57	
17	15061	54	
18	15189	53	
19	13767	52	
20	15039	52	

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과			
작업 정보	결과	시각화	JSON
실행 세부정보	실행 그래프		
명	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	
11	12357	2708	
12	12358	242	
13	12359	1599	
14	12360	1156	
15	12361	90	
16	12362	2180	
17	12363	408	
18	12364	1499	
19	12365	173	
20	12367	172	

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 `smiling-mark-468901-i5.modulabs_project.user_rf` AS
```

```
-- (1) 전체 거래 건수 계산
```

```
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `smiling-mark-468901-i5.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
),
```

```
-- (2) 구매한 아이템 총 수량 계산
```

```
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM `smiling-mark-468901-i5.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
```

```
-- user_r와 결합
```

```
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.recency_days AS recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN `smiling-mark-468901-i5.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보				
결과				
실행 세부정보				
실행 그래프				
이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.				

명	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	14569	1	79	1
3	15520	1	314	1
4	13298	1	96	1
5	13436	1	76	1
6	14204	1	72	2
7	15471	1	256	2
8	15195	1	1404	2
9	12650	1	250	3
10	16528	1	171	3
11	17914	1	457	3
12	14578	1	240	3
13	15318	1	642	3
14	15992	1	17	3
15	16569	1	93	3
16	12478	1	233	3
17	12442	1	181	3
18	17383	1	148	4
19	18015	1	157	4
20	12367	1	172	4
21	14219	1	78	4
22	15097	1	170	4
23	13790	1	748	4
24	16597	1	184	4

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362명)

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `smiling-mark-468901-i5.modulabs_project.data`
WHERE CustomerID IS NOT NULL
  AND Quantity > 0      -- 반품/취소(음수 수량) 제외
GROUP BY CustomerID
ORDER BY user_total DESC;

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
명	CustomerID	user_total			
6	12415	124864.5			
7	14156	116560.1			
8	17511	91062.4			
9	12346	77183.6			
10	16029	72708.1			
11	16684	66653.6			
12	13694	65039.6			
13	15311	60632.8			
14	13089	58762.1			
15	17949	58030.5			
16	15769	56352.7			
17	15061	54534.1			
18	14096	53058.4			
19	14298	51527.3			
20	14088	50491.8			
21	15749	44534.3			
22	12931	42056.0			

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4304행) |

#### • 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 `smiling-mark-468901-i5.modulabs_project.user_rfm` AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ROUND(ut.user_total, 1) AS user_total,
  ROUND(SAFE_DIVIDE(ut.user_total, rf.purchase_cnt), 1) AS user_average
FROM `smiling-mark-468901-i5.modulabs_project.user_rf` rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    SUM(CASE WHEN Quantity > 0 THEN Quantity * UnitPrice ELSE 0 END) AS user_total
  FROM `smiling-mark-468901-i5.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.         </div>			

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	794.5
2	14569	1	79	1	227.4	227.4
3	15520	1	314	1	343.5	343.5
4	13436	1	76	1	196.9	196.9
5	13298	1	96	1	360.0	360.0
6	15471	1	256	2	454.5	454.5
7	14204	1	72	2	150.6	150.6
8	15195	1	1404	2	3861.0	3861.0
9	14578	1	240	3	168.6	168.6
10	15992	1	17	3	42.0	42.0
11	12442	1	181	3	144.1	144.1
12	12650	1	250	3	242.4	242.4
13	16569	1	93	3	124.2	124.2
14	17914	1	457	3	329.4	329.4
15	16528	1	171	3	244.4	244.4
16	12478	1	233	3	546.0	546.0
17	15318	1	642	3	312.6	312.6
18	18015	1	157	4	120.0	120.0
19	14219	1	78	4	89.9	89.9
20	17383	1	148	4	193.4	193.4
21	16597	1	184	4	90.0	90.0
22	15097	1	170	4	248.1	248.1
23	12367	1	172	4	150.9	150.9
24	13790	1	748	4	348.8	348.8

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

## RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```
SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.user_rfm`
LIMIT 50;
```

```
--고유한 유저 수는 user_rfm이 고객 1명당 1행이므로 COUNT
SELECT COUNT(*) AS unique_users
FROM `smiling-mark-468901-i5.modulabs_project.user_rfm`;
```

[결과 이미지를 넣어주세요]

쿼리 결과						
작업 정보	결과	시각화	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	14569	1	79	1	227.4	227.4
3	15520	1	314	1	343.5	343.5
4	13436	1	76	1	196.9	196.9
5	13298	1	96	1	360.0	360.0
6	15471	1	256	2	454.5	454.5
7	14204	1	72	2	150.6	150.6
8	15195	1	1404	2	3861.0	3861.0
9	14578	1	240	3	168.6	168.6
10	15992	1	17	3	42.0	42.0
11	12442	1	181	3	144.1	144.1
12	12650	1	250	3	242.4	242.4
13	16569	1	93	3	124.2	124.2
14	17914	1	457	3	329.4	329.4
15	16528	1	171	3	244.4	244.4
16	12478	1	233	3	546.0	546.0
17	15318	1	642	3	312.6	312.6
18	18015	1	157	4	120.0	120.0
19	14219	1	78	4	89.9	89.9
20	17383	1	148	4	193.4	193.4

페이지당 결과 수: 50 1 ~ 50 (전체 50행)

← 쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
unique_users	4362				

## 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 smiling-mark-468901-i5.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM smiling-mark-468901-i5.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM smiling-mark-468901-i5.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

명	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...
1	12713	1	505	0	794.5	794.5	37
2	13298	1	96	1	360.0	360.0	2
3	13436	1	76	1	196.9	196.9	12
4	15520	1	314	1	343.5	343.5	18
5	14569	1	79	1	227.4	227.4	10
6	14204	1	72	2	150.6	150.6	36
7	15195	1	1404	2	3861.0	3861.0	1
8	15471	1	256	2	454.5	454.5	67
9	12478	1	233	3	546.0	546.0	35
10	15992	1	17	3	42.0	42.0	3
11	12650	1	250	3	242.4	242.4	19
12	16528	1	171	3	244.4	244.4	17
13	16569	1	93	3	124.2	124.2	5
14	14578	1	240	3	168.6	168.6	24
15	15318	1	642	3	312.6	312.6	33
16	17914	1	457	3	329.4	329.4	72
17	12442	1	181	3	144.1	144.1	11
18	17383	1	148	4	193.4	193.4	49
19	14219	1	78	4	89.9	89.9	7
20	15097	1	170	4	248.1	248.1	25
21	18015	1	157	4	120.0	120.0	46
22	16597	1	184	4	90.0	90.0	7
23	12367	1	172	4	150.9	150.9	10
24	13790	1	748	4	348.8	348.8	45

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362명)

## 2. 평균 구매 주기

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

```
CREATE OR REPLACE TABLE `smiling-mark-468901-i5.modulabs_project.user_data` AS
WITH purchase_intervals AS (
  -- (2) 고객별 평균 구매 간격(일)
  SELECT
    CustomerID,
    COALESCE(ROUND(AVG(interval_), 2), 0) AS average_interval
  FROM (
    -- (1) 연속 구매 간 일수
    SELECT
      CustomerID,
      DATE_DIFF(
        DATE(InvoiceDate),
        LAG(DATE(InvoiceDate)) OVER (PARTITION BY CustomerID ORDER BY DATE(InvoiceDate)),
        DAY
      ) AS interval_
    FROM `smiling-mark-468901-i5.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  WHERE interval_ IS NOT NULL
  GROUP BY CustomerID
)
```

```
SELECT
  u.* EXCEPT (average_interval),  -- u의 average_interval 제거
  pi.average_interval              -- pi의 값을 최종 컬럼으로 사용
FROM `smiling-mark-468901-i5.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
  ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

명	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod	average_inter	
1	16323	1	50	196	207.5	207.5	1	null	
2	13703	1	10	318	99.5	99.5	1	null	
3	13747	1	8	373	79.6	79.6	1	null	
4	16144	1	16	246	175.2	175.2	1	null	
5	14576	1	12	372	35.4	35.4	1	null	
6	16765	1	4	294	34.0	34.0	1	null	
7	12814	1	48	101	85.9	85.9	1	null	
8	15657	1	24	22	30.0	30.0	1	null	
9	14351	1	12	164	51.0	51.0	1	null	
10	15510	1	2	330	250.0	250.0	1	null	
11	15668	1	72	217	76.3	76.3	1	null	
12	18174	1	50	7	104.0	104.0	1	null	
13	16881	1	600	66	432.0	432.0	1	null	
14	15118	1	1440	134	244.8	244.8	1	null	
15	13366	1	144	50	56.2	56.2	1	null	
16	17443	1	504	219	534.2	534.2	1	null	
17	15313	1	25	110	52.0	52.0	1	null	
18	16454	1	2	64	5.9	5.9	1	null	
19	17331	1	16	123	175.2	175.2	1	null	
20	13120	1	12	238	30.6	30.6	1	null	
21	18184	1	60	15	49.8	49.8	1	null	
22	16344	1	18	158	101.1	101.1	1	0.0	
23	17948	1	144	147	358.6	358.6	1	null	
24	16257	1	1	176	21.9	21.9	1	null	

페이지당 결과 수: 50 ▼ 1 ~ 50 (전체 4362행)

### 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 `smiling-mark-468901-i5.modulabs_project.user_data` AS
```

```
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT IF(InvoiceNo LIKE 'C%', InvoiceNo, NULL)) AS cancel_frequency
  FROM `smiling-mark-468901-i5.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
```

```
SELECT
  u.*,
  t.* EXCEPT (CustomerID),
  ROUND(SAFE_DIVIDE(t.cancel_frequency, t.total_transactions) * 100, 2) AS cancel_rate
FROM `smiling-mark-468901-i5.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
  ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...	total_transac...	cancel_freque...	cancel_rate
1	16737	1	288	53	417.6	417.6	1	null	1	0	0.0
2	17394	1	241	219	203.9	203.9	4	0.0	1	0	0.0
3	12659	1	104	29	73.7	73.7	4	0.0	1	0	0.0
4	12756	1	144	86	112.1	112.1	4	0.0	1	0	0.0
5	14373	1	17	358	76.6	76.6	4	0.0	1	0	0.0
6	14890	1	51	253	125.7	125.7	7	0.0	1	0	0.0
7	16665	1	59	362	135.4	135.4	9	0.0	1	0	0.0
8	16179	1	99	60	215.8	215.8	11	0.0	1	0	0.0
9	15917	1	42	247	123.1	123.1	12	0.0	1	0	0.0
10	17732	1	93	372	304.0	304.0	18	0.0	1	0	0.0
11	13774	1	192	81	345.0	345.0	22	0.0	1	0	0.0
12	13244	1	38	92	121.7	121.7	27	0.0	1	0	0.0
13	15947	1	898	82	1708.2	1708.2	29	0.0	1	0	0.0
14	16461	1	48	177	142.9	142.9	31	0.0	1	0	0.0
15	13355	1	314	122	674.7	674.7	32	0.0	1	0	0.0
16	12514	1	440	267	1017.7	1017.7	51	0.0	1	0	0.0
17	16445	1	111	33	230.5	230.5	62	0.0	1	0	0.0
18	15471	1	256	2	454.5	454.5	67	0.0	1	0	0.0
19	14093	1	391	12	433.9	433.9	95	0.0	1	0	0.0
20	14424	1	48	17	322.1	322.1	1	null	1	0	0.0
21	13185	1	12	267	71.4	71.4	1	null	1	0	0.0
22	14199	1	28	218	185.4	185.4	2	0.0	1	0	0.0
23	12401	1	10	303	69.3	69.3	4	0.0	1	0	0.0

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...	total_transac...	cancel_freque...	cancel_rate
1	16737	1	288	53	417.6	417.6	1	null	1	0	0.0
2	17394	1	241	219	203.9	203.9	4	0.0	1	0	0.0
3	12659	1	104	29	73.7	73.7	4	0.0	1	0	0.0
4	12756	1	144	86	112.1	112.1	4	0.0	1	0	0.0
5	14373	1	17	358	76.6	76.6	4	0.0	1	0	0.0
6	14890	1	51	253	125.7	125.7	7	0.0	1	0	0.0
7	16665	1	59	362	135.4	135.4	9	0.0	1	0	0.0
8	16179	1	99	60	215.8	215.8	11	0.0	1	0	0.0
9	15917	1	42	247	123.1	123.1	12	0.0	1	0	0.0
10	17732	1	93	372	304.0	304.0	18	0.0	1	0	0.0
11	13774	1	192	81	345.0	345.0	22	0.0	1	0	0.0
12	13244	1	38	92	121.7	121.7	27	0.0	1	0	0.0
13	15947	1	898	82	1708.2	1708.2	29	0.0	1	0	0.0
14	16461	1	48	177	142.9	142.9	31	0.0	1	0	0.0
15	13355	1	314	122	674.7	674.7	32	0.0	1	0	0.0
16	12514	1	440	267	1017.7	1017.7	51	0.0	1	0	0.0
17	16445	1	111	33	230.5	230.5	62	0.0	1	0	0.0
18	15471	1	256	2	454.5	454.5	67	0.0	1	0	0.0
19	14093	1	391	12	433.9	433.9	95	0.0	1	0	0.0
20	14424	1	48	17	322.1	322.1	1	null	1	0	0.0
21	13185	1	12	267	71.4	71.4	1	null	1	0	0.0
22	14199	1	28	218	185.4	185.4	2	0.0	1	0	0.0
23	12401	1	10	303	69.3	69.3	4	0.0	1	0	0.0

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

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

```
SELECT *
FROM `smiling-mark-468901-i5.modulabs_project.user_data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]

←

쿼리 결과

결과 저장

다음에서 열거

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프						
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16148	1	72	296	76.3	76.3	1	null	1	0	0.0
2	15992	1	17	3	42.0	42.0	3	0.0	1	0	0.0
3	15063	1	56	213	370.8	370.8	5	0.0	1	0	0.0
4	18249	1	128	17	95.3	95.3	8	0.0	1	0	0.0
5	17698	1	93	260	154.9	154.9	8	0.0	1	0	0.0
6	15683	1	131	117	193.8	193.8	10	0.0	1	0	0.0
7	18066	1	108	75	318.0	318.0	10	0.0	1	0	0.0
8	12622	1	68	232	162.1	162.1	10	0.0	1	0	0.0
9	17030	1	146	63	146.9	146.9	11	0.0	1	0	0.0
10	13858	1	75	57	216.2	216.2	11	0.0	1	0	0.0
11	12732	1	110	179	218.3	218.3	11	0.0	1	0	0.0
12	14804	1	121	8	353.3	353.3	12	0.0	1	0	0.0
13	13065	1	74	373	205.9	205.9	14	0.0	1	0	0.0
14	13328	1	680	316	1308.5	1308.5	17	0.0	1	0	0.0
15	18120	1	31	214	111.2	111.2	17	0.0	1	0	0.0

페이지당 결과 수: 50

1 - 50 (전체 100행)

이전

다음

페이지당 결과 수: 50 1 ~ 50 (전체 100행) |< >

## 회고

[회고 내용을 작성해주세요]

Keep :

Problem :

Try :