

# 고객을 세그멘테이션하자 [프로젝트]

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
select * from modulabs_project.data
limit 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과									
작업 정보									
결과									
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 ...	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.LIGH...	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 D...	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 modulabs_project.data
;
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보		
결과		
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 modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과									
작업 정보									
결과									
차트									
JSON									
실행 세부정보									
실행 그래프									
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 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 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 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 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 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 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 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 modulabs_project.data
;
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보 <b>결과</b> 차트   JSON   실행 세부정보		
행	column_name ▼	missing_percentage
1	UnitPrice	0.0
2	CustomerID	24.93
3	InvoiceNo	0.0
4	InvoiceDate	0.0
5	Country	0.0
6	Description	0.27
7	Quantity	0.0
8	StockCode	0.0

## 결측치 처리 전략

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

```
select DISTINCT(description)
from modulabs_project.data
where StockCode = '85123A' ;
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보 <b>결과</b> 차트   JSON   실행		
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 modulabs_project.data
where customerID is null;
```

[결과 이미지를 넣어주세요]

쿼리 결과			
작업 정보 <b>결과</b> 실행 세부정보   실행 그래프			
<b>i</b> 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 COUNT(*) AS duplicate_row_count
FROM (
  SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, COUNT(*) AS cnt
  FROM modulabs_project.data
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
) AS duplicates;
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보 <b>결과</b> 차트   J:		
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

```
# [[YOUR QUERY]];
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT DISTINCT *
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

```
select count(*)
from modulabs_project.data;
```

쿼리 결과		
작업 정보 <b>결과</b> 차트		
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 modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON
행	f0_		
1	22190		

- 고유한 **InvoiceNo**를 앞에서부터 100개를 출력하기

```
# [[YOUR QUERY]]
select distinct(invoiceNo)
from modulabs_project.data
limit 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
1	541431				
2	C541433				
3	537626				
4	542237				
5	549222				
6	556201				
7	562032				
8	573511				
9	581180				
10	539318				
11	541998				
12	548955				
13	568172				
14	577609				
15	543037				
16	544156				

페이지당 결과 수: 50 1 - 50 (전체 100행) |< < > >

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

```
select *
from `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	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STOR...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
6	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
7	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
8	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
9	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
10	C547388	84050	PINK HEART SHAPE EGG FRYL...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
11	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
12	C549955	22839	3 TIER CAKE TIN GREEN AND...	-2	2011-04-13 13:38:00 UTC	14.95	12359	Cyprus
13	C549955	22666	RECIPES BOX PANTRY YELLOW ...	-2	2011-04-13 13:38:00 UTC	2.95	12359	Cyprus
14	C580165	22797	CHEST OF DRAWERS GINGHA...	-2	2011-12-02 11:21:00 UTC	16.95	12359	Cyprus
15	C580165	22720	SET OF 3 CAKE TINS PANTRY ...	-1	2011-12-02 11:21:00 UTC	4.95	12359	Cyprus
16	C580165	23245	SET OF 3 REGENCY CAKE TINS	-2	2011-12-02 11:21:00 UTC	4.95	12359	Cyprus

페이지당 결과 수: 50 1 - 50 (전체 100행) |< >

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

```
select ROUND(SUM(CASE WHEN quantity < 0 THEN 1 ELSE 0 END)/COUNT(*) * 100, 1)
from `modulabs_project.data`
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 차트 JS

행	f0_	
1	2.2	

## StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
# [[YOUR QUERY]]
select DISTINCT(StockCode)
from `modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	StockCode
1	23166
2	22805
3	22195
4	22773
5	21064
6	851678
7	71477
8	85116
9	22774
10	849978
11	22375
12	85175

페이지당 결과 수: 50 1 - 50 (전체 3684행) |<

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 **StockCode** 별 등장 빈도를 출력하기

- 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `modulabs_project.data`
Group by StockCode
ORDER BY sell_cnt DESC
limit 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
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 `modulabs_project.data`
)
WHERE number_count = 0 or number_count = 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
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(COUNTIF(number_count = 0 OR number_count = 1) / COUNT(*) * 100, 2) as percentage
from (
  SELECT *,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `modulabs_project.data`
);
```


[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보 <b>결과</b> 차트		
행	percentage	
1	0.48	

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `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 `modulabs_project.data`
  ) sub
  WHERE number_count = 0 or number_count = 1
);
```

[결과 이미지를 넣어주세요]

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

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM `modulabs_project.data`
group by description
order by description_cnt DESC
limit 30;
```

[결과 이미지를 넣어주세요]



## 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
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 CAK...		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	

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
from `modulabs_project.data`
where Description in ('High Resolution Image','Next Day Carriage');
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
	<p><b>i</b> 이 문으로 data의 행 83개가 삭제되었습니다.</p>		

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

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

## 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 `modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과				
작업 정보 <b>결과</b> 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 `modulabs_project.data`
WHERE unitprice = 0.0;
```

[결과 이미지를 넣어주세요]

쿼리 결과					
작업 정보 <b>결과</b> 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 `modulabs_project.data` AS
SELECT *
from `modulabs_project.data`
where unitprice != 0.0;
```

[결과 이미지를 넣어주세요]

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

## 11-7. RFM 스코어

### Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay
from `modulabs_project.data`
```

[결과 이미지를 넣어주세요]

#### 쿼리 결과

작업 정보	결과	차트
행	InvoiceDay	
1	2011-01-18	
2	2011-01-18	
3	2010-12-07	
4	2010-12-07	
5	2010-12-07	
6	2010-12-07	
7	2010-12-07	
8	2010-12-07	
9	2010-12-07	
10	2010-12-07	

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  max(date(InvoiceDate)) as most_recent_date
from `modulabs_project.data`
```

[결과 이미지를 넣어주세요]

#### 쿼리 결과

작업 정보	결과	차트
행	most_recent_date	
1	2011-12-09	

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  max(date(InvoiceDate)) as most_recent_date
FROM `modulabs_project.data`
group by customerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보	결과	차트
JSON		
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

- 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]

쿼리 결과		
작업 정보	결과	차트
JSON		
행	CustomerID	recency
1	12413	66
2	12436	99
3	12573	227
4	12936	17
5	13006	56
6	13115	8
7	13136	50
8	13248	124
9	13283	59
10	13366	50

- 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 **user\_r** 이라는 이름의 테이블로 저장하기

```
CREATE OR REPLACE TABLE `radiant-mason-456102-b4.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 `radiant-mason-456102-b4.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.

```
select *  
from `radiant-mason-456102-b4.modulabs_project.user_r`;
```

접근성 옵션을 보

### 리 결과

결과 저장   

작업 정보   **결과**   차트   JSON   실행 세부정보   실행 그래프

	CustomerID ▾	recency ▾
1	15694	0
2	14446	0
3	13777	0
4	15910	0
5	14422	0
6	12662	0
7	14051	0
8	16705	0
9	12526	0
10	13069	0

페이지당 결과 수: 50 ▾   1 - 50 (전체 4362행)

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT  
  CustomerID,  
  count(DISTINCT InvoiceNo) AS purchase_cnt  
FROM `radiant-mason-456102-b4.modulabs_project.data`  
group by CustomerID;
```

[결과 이미지를 넣어주세요]

### 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부
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 `radiant-mason-456102-b4.modulabs_project.data`
group by CustomerID;
```

[결과 이미지를 넣어주세요]

### 쿼리 결과

작업 정보	결과	차트	JSON
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 `radiant-mason-456102-b4.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    count(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `radiant-mason-456102-b4.modulabs_project.data`
  group by CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
```

```

SELECT
  CustomerID,
  sum(Quantity) AS item_cnt
FROM `radiant-mason-456102-b4.modulabs_project.data`
group by CustomerID
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.regency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN `radiant-mason-456102-b4.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.

295 | `select * from radiant-mason-456102-b4.modulabs_project.user_rf;` [접근성 옵션을 보러](#)

쿼리 결과 [결과 저장](#) [다시 실행](#)

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	14569	1	79	1
3	13298	1	96	1
4	15520	1	314	1
5	13436	1	76	1
6	15471	1	256	2
7	15195	1	1404	2
8	14204	1	72	2
9	14578	1	240	3
10	15992	1	17	3
11	16560	1	83	2

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
  CustomerID,
  ROUND(sum(Quantity*UnitPrice),1) AS user_total
FROM `radiant-mason-456102-b4.modulabs_project.data`
group by CustomerID;

```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	차트
작업 정보			결과	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		
6	12352	1265.4		
7	12353	89.0		
8	12354	1079.4		
9	12355	459.4		
10	12356	2487.4		
11	12357	5007.7		

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 radiant-mason-456102-b4.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, 2) AS user_average
FROM radiant-mason-456102-b4.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(sum(Quantity*UnitPrice),1) AS user_total
  FROM `radiant-mason-456102-b4.modulabs_project.data`
  group by CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

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

RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```
# [[YOUR QUERY]];
select * from radiant-mason-456102-b4.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]



```

323 select * from radiant-mason-456102-b4.modulabs_project.user_rfm;
324

```

점근성 옵션을 보려면 Alt+F1 키를

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
1	12713	1	505	0	794.5	794.5
2	15520	1	314	1	343.5	343.5
3	14569	1	79	1	227.4	227.4
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	16569	1	93	3	124.2	124.2
10	12442	1	181	3	144.1	144.1
11	15210	1	640	3	310.6	310.6

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 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]

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

**i** 이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

338 `select * from radiant-mason-456102-b4.modulabs_project.user_data;` 점근성 옵션을 보려면 Alt+F1 키를 누르세요

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_p
1	16738	1	3	297	3.8	3.8	
2	17331	1	16	123	175.2	175.2	
3	16990	1	100	218	179.0	179.0	
4	15070	1	36	372	106.2	106.2	
5	16881	1	600	66	432.0	432.0	
6	14679	1	-1	371	-2.5	-2.5	
7	16454	1	2	64	5.9	5.9	

페이지당 결과 수: 50 1 - 50 (전체 4362행) |< < > >|


## 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 radiant-mason-456102-b4.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
      radiant-mason-456102-b4.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM radiant-mason-456102-b4.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 이름이 user_data인 테이블이 교체되었습니다.         </div>			

```

365 SELECT *
366 FROM radiant-mason-456102-b4.modulabs_project.user_data;
367

```

접근성 옵션을 보려면 Alt+F1 키를

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	average_interval
1	16765	1	4	294	34.0	34.0	1	0.0
2	15488	1	72	92	76.3	76.3	1	0.0
3	15316	1	100	326	165.0	165.0	1	0.0
4	13747	1	8	373	79.6	79.6	1	0.0
5	13307	1	4	120	15.0	15.0	1	0.0
6	16995	1	-1	372	-1.3	-1.3	1	0.0
7	15753	1	144	304	79.2	79.2	1	0.0
8	16138	1	-1	368	-8.0	-8.0	1	0.0

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 radiant-mason-456102-b4.modulabs_project.user_data AS
```

```
WITH TransactionInfo AS (
```

```
SELECT
  CustomerID,
  COUNT(*) AS total_transactions,
  COUNTIF(LEFT(InvoiceNo, 1) = 'C') AS cancel_frequency
FROM radiant-mason-456102-b4.modulabs_project.data
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
)
```

```
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / NULLIF(t.total_transactions, 0), 2) AS cancel_rate
FROM `radiant-mason-456102-b4.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과	
작업 정보	결과
<div> <div></div> <div>이 문으로 이름이 user_data인 테이블이 교체되었습니다.</div> </div>	

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

```
# [[YOUR QUERY]];
SELECT *
FROM radiant-mason-456102-b4.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

명	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate	
1	16024	1	377	12	245.7	245.7	60	0.0	61	0	0.0	
2	12847	1	873	22	843.6	843.6	72	0.0	84	0	0.0	
3	15069	1	520	179	1109.5	1109.5	70	0.0	70	0	0.0	
4	12349	1	630	18	1457.6	1457.6	72	0.0	72	0	0.0	
5	12890	1	256	24	376.7	376.7	78	0.0	79	0	0.0	
6	17914	1	457	3	329.4	329.4	72	0.0	75	0	0.0	
7	16222	1	325	318	773.6	773.6	120	0.0	122	0	0.0	
8	14382	1	1231	26	615.5	615.5	121	0.0	126	0	0.0	

페이지당 결과 수: 50

1 - 50 (전체 4362명)

<

>

<>

페이지당 결과 수: 50 1 - 50 (전체 4362명) |< > >|

## 회고

[회고 내용을 작성해주세요]

Keep : 팀원분들의 어려운 점을 물어보고 도움을 줄 수 있는 만큼 도와드렸다.

Problem : ChatGPT 의 도움으로 문법 오류를 찾아낼 수 있었다.

Try : 문법이나 용어, 개념을 정리해야겠다.