

# Main Quest 1 (권미경)

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM `decoded-indexer-439402-q8.modulabs_project.data`  
LIMIT 10
```

[결과 이미지를 넣어주세요]

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

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

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [MY QUERY STRARETY] COUNT(*) AS row_count  
  
SELECT COUNT(*) AS rwo_count  
FROM `decoded-indexer-439402-q8.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 데이터 탐색

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

행	rwo_count ▼
1	541909

## 데이터 수 세기

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

```
# [MY QUERY STRARETY] COUNTN(개별 컬럼) AS COUNT_개별컬럼명
```

```
SELECT
    COUNT(InvoiceNo) AS COUNT_InvoiceNo,
    COUNT(StockCode) AS COUNT_StockCode,
    COUNT(Description) AS COUNT_Description,
    COUNT(Quantity) AS COUNT_Quantity,
    COUNT(InvoiceDate) AS InvoiceDate,
    COUNT(UnitPrice) AS COUNT_UnitPrice,
    COUNT(CustomerID) AS COUNT_CustomerID,
    COUNT(Country) AS COUNT_Country
FROM decoded-indexer-439402-q8.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과									<a href="#">결과 저장</a>
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프				
행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country	
1	541909	541909	540455	541909	541909	541909	406829	541909	

## 11-4. 데이터 전처리 방법(1): 결측치 제거

### 컬럼 별 누락된 값의 비율 계산

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

```
# [MY QUERY STRARETY]
-- CASE WHEN TEHN 1 ELSE 0을 사용하여 InvoiceNo의 결측치 비율 계산
-- ROUND(number, decimal_places) :소수점 자리수로 반올림 함수
-- UNION ALL

SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.modulabs_project.data;

```

[결과 이미지를 넣어주세요]

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

## 결측치 처리 전략

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

```
# [MY QUERY STRARETY] WERER 절 사용
# 오류수정 STRING 작은따옴표 교체('85123A')/ 결과이미지는 결측치 제거 후 추출결과

SELECT Description
FROM `decoded-indexer-439402-q8.modulabs_project.data`
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

쿼리 결과		결과 저장	데이터 탐색	
작업 정보	결과	차트	JSON	실행 세부정보
실행 그래프				
행	Description			
1	WHITE HANGING HEART T-LIG...			
2	WHITE HANGING HEART T-LIG...			
3	WHITE HANGING HEART T-LIG...			
4	WHITE HANGING HEART T-LIG...			
5	WHITE HANGING HEART T-LIG...			
6	WHITE HANGING HEART T-LIG...			
7	WHITE HANGING HEART T-LIG...			

## 결측치 처리

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

```
# [MY QUERY STRARETY] 결측치 제거 : DELETER FROM 구문, WHERE 절 사용

DELETE FROM `decoded-indexer-439402-q8.modulabs_project.data`
WHERE CustomerID IS NULL
OR Description IS NULL ;
```

[결과 이미지를 넣어주세요]

쿼리 결과

 결과 저장

작업 정보

결과

실행 세부정보

실행 그래프

 이 문으로 data의 행 133,626개가 삭제되었습니다.

## 11-5. 데이터 전처리(2): 중복값 처리

### 컬럼 별 누락된 값의 비율 계산

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

```
# [MY QUERY STRARETY]
-- CASE WHEN TEHN 1 ELSE 0을 사용하여 InvoiceNo의 결측치 비율 계산
-- ROUND(number, decimal_places) :소수점 자리수로 반올림 함수
-- UNION ALL

SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percenta
ge
FROM decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.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 decoded-indexer-439402-q8.modulabs_project.data;

```

[결과 이미지를 넣어주세요]

쿼리 결과

 결과 저장 ▾

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

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

## 중복값 확인

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

```
# [MY QUERY STRARETY] WITH AS, GROUP BY, HAVING, COUNT>1

With duplicate_groups AS (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS row_count
  FROM
    `decoded-indexer-439402-q8.modulabs_project.data`
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  HAVING
    row_count > 1
)

SELECT COUNT(*) AS duplicate_count
FROM duplicate_groups
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장
작업 정보	결과	차트	JSON
실행 세부정보	실행 그래프		
행	duplicate_count		
1	4837		

## 중복값 처리

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

```
# [MY QUERY STRARETY] 중복값 제거 : CREATE OR REPLACE TABLE AS, SELECT DISTINCT *

CREATE OR REPLACE TABLE `decoded-indexer-439402-q8.modulabs_project.distinct_data` AS
SELECT DISTINCT *
FROM `decoded-indexer-439402-q8.modulabs_project.data`;

SELECT COUNT(*)
FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`
LIMIT 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장
작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 distinct_data인 새 테이블이 생성되었습니다.			

  

쿼리 결과			결과 저장
작업 정보	결과	차트	JSON
실행 세부정보	실행 그래프		
행	f0_		
1	401604		

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
# [MY QUERY STRARETY] COUNT(DISTINCT 컬럼명)

SELECT
COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	데이터 탐색
작업 정보	결과	차트	JSON	실행 세부정보
실행 그래프				
행	unique_invoice_coun			
1	22190			

- 고유한 **InvoiceNo** 를 앞에서부터 100개를 출력하기

```
# [MY QUERY STRARETY] SELECT DISTINCT
```

```
SELECT DISTINCT InvoiceNo
FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	
1	574301	
2	C575531	
3	557305	
4	543008	
5	549735	

페이지당 결과 수:

50

1 - 50 (전체 100행)

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

```
# [MY QUERY STRARETY] WHERE LIKE
```

```
SELECT *
FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity
1	C575531	22960	JAM MAKING SET WITH JARS	
2	C558080	22840	ROUND CAKE TIN VINTAGE RED	
3	C558080	22847	BREAD BIN DINER STYLE IVORY	
4	C554983	47590B	PINK HAPPY BIRTHDAY BUNTI...	
5	C554983	47590A	PINK HAPPY BIRTHDAY BUNTI...	

페이지당 결과 수: 50

1 - 50 (전체 100행)

<<

<

>

>>

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

```
# 정답은 2.2% 라던데..... ????? 미완결 추후 보정 ^^
```

```
SELECT
ROUND(COUNT(DISTINCT CASE WHEN InvoiceNo Like 'C%' THEN InvoiceNo END) / COUNT(DISTINCT InvoiceNo) * 100,1) AS perrcentage_c_invoices
FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`
```

[결과 이미지를 넣어주세요]



쿼리 결과			결과 저장	데이터 탐색	
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	percentage_c_invoic				
1	16.5				

## StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
# [MY QUERY STRARETY] COUNT(DISTINCT 컬럼명)

SELECT
    COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	데이터 탐색	
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	unique_stockcode_c				
1	3684				

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

```
# [MY QUERY STRARETY] COUNT(*), GROUP BY, ORDER BY

SELECT StockCode, COUNT(*) AS sell_cnt
FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과				결과 저장	데이터 탐색	
작업 정보				결과	차트	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

```
# [MY QUERY STRARETY] FROM (SUBQUERY : LENGTH(REGEXP_REPLACE), WHERE

SELECT DISTINCT StockCode, number_cnt
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(stockcode, r'[0-9]', ''))AS number_cnt
  FROM `decoded-indexer-439402-q8.modulabs_project.distinct_data`
)
WHERE number_cnt < 2 ;
```

[결과 이미지를 넣어주세요]

쿼리 결과				결과 저장
작업 정보				결과
				차트
				JSON
				실행 세부정보
				실행 그래프
행	StockCode	number_cnt		
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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM project_name.modulabs_project.data
)
WHERE # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM project_name.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    # [[YOUR QUERY]]
  );
```

[결과 이미지를 넣어주세요]

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM project_name.modulabs_project.data
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM project_name.modulabs_project.data
WHERE
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  # [[YOUR QUERY]] AS Description
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

## UnitPrice 살펴보기

- UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```
SELECT # [[YOUR QUERY]] AS min_price, # [[YOUR QUERY]] AS max_price, # [[YOUR QUERY]] AS avg_price
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기

```
SELECT # [[YOUR QUERY]] AS cnt_quantity, # [[YOUR QUERY]] AS min_quantity, # [[YOUR QUERY]] AS max_quantity, # [[YOUR QUERY]] AS avg_quantity
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT *
```

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

[결과 이미지를 넣어주세요]

## 11-7. RFM 스코어

### Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT # [[YOUR QUERY]] AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

- 가장 최근 구매 일자를 **MAX()** 함수로 찾아보기

```
SELECT
    # [[YOUR QUERY]] AS most_recent_date,
    # [[YOUR QUERY]] AS InvoiceDay,
    *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

- 유저 별로 가장 큰 **InvoiceDay**를 찾아서 가장 최근 구매일로 저장하기

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

[결과 이미지를 넣어주세요]

- 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 **InvoiceNo**의 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 계산과 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
- 1. `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)
  - 군 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_inter
val
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
        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
- 취소 빈도(cancel\_frequency) : 고객 별로 취소한 거래의 총 횟수
  - 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 project_name.modulabs_project.user_data AS

WITH TransactionInfo AS (
    SELECT
        CustomerID,
        # [[YOUR QUERY]] AS total_transactions,
        # [[YOUR QUERY]] AS cancel_frequency
    FROM project_name.modulabs_project.data
    # [[YOUR QUERY]]
)

SELECT u.*, t.* EXCEPT(CustomerID), # [[YOUR QUERY]] AS cancel_rate
FROM `project_name.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON # [[YOUR QUERY]];
```

[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]

## 회고

[회고 내용을 작성해주세요]

