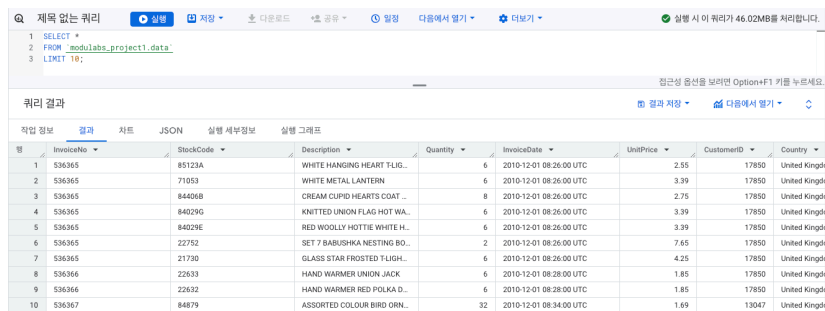


11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

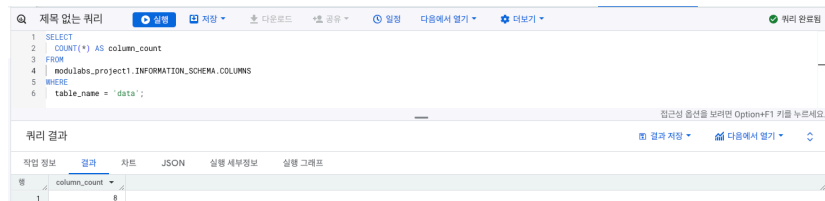
```
SELECT *  
FROM modulabs_project1.data  
LIMIT 10;
```



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 Kingd
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingd
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingd
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingd
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingd
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingd
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingd
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingd
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingd
10	536367	84879	ASSORTED COLOUR BIRD DRN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingd

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT  
COUNT(*) AS column_count  
FROM  
modulabs_project1.INFORMATION_SCHEMA.COLUMNS  
WHERE  
table_name = 'data';
```



행	column_count
1	8

데이터 수 세기

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

```
WITH columns AS(  
SELECT  
column_name  
FROM  
modulabs_project1.INFORMATION_SCHEMA.COLUMNS  
WHERE  
table_name = 'data'  
)
```

SELECT

```
'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS non_null_count FROM `modulabs_project1.data` UNION ALL
SELECT 'StockCode', COUNT(StockCode) FROM `modulabs_project1.data` UNION ALL
SELECT 'Description', COUNT(Description) FROM `modulabs_project1.data` UNION ALL
SELECT 'Quantity', COUNT(Quantity) FROM `modulabs_project1.data` UNION ALL
SELECT 'InvoiceDate', COUNT(InvoiceDate) FROM `modulabs_project1.data` UNION ALL
SELECT 'UnitPrice', COUNT(UnitPrice) FROM `modulabs_project1.data` UNION ALL
SELECT 'CustomerID', COUNT(CustomerID) FROM `modulabs_project1.data` UNION ALL
SELECT 'Country', COUNT(Country) FROM `modulabs_project1.data` ;
```

제목 없는 쿼리 실행 다운로드 공유 일정 다음에서 열기 더보기 저장 실행 시 이 쿼리가 56.02MB를 처리합니다.

```
1 WITH columns AS(
2   SELECT
3     column_name
4   FROM
5     modulabs_project1.INFORMATION_SCHEMA.COLUMNS
6   WHERE
7     table_name = 'data'
8 )
9
10 SELECT
11   'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS non_null_count FROM `modulabs_project1.data` UNION ALL
12   SELECT 'StockCode', COUNT(StockCode) FROM `modulabs_project1.data` UNION ALL
13   SELECT 'Description', COUNT(Description) FROM `modulabs_project1.data` UNION ALL
14   SELECT 'Quantity', COUNT(Quantity) FROM `modulabs_project1.data` UNION ALL
15   SELECT 'InvoiceDate', COUNT(InvoiceDate) FROM `modulabs_project1.data` UNION ALL
16   SELECT 'UnitPrice', COUNT(UnitPrice) FROM `modulabs_project1.data` UNION ALL
17   SELECT 'CustomerID', COUNT(CustomerID) FROM `modulabs_project1.data` UNION ALL
18   SELECT 'Country', COUNT(Country) FROM `modulabs_project1.data` ;
--
```

최근성 옵션을 보려면 Option+F1 키를 누르세요.

쿼리 결과 결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	column_name	non_null_count			
1	Country	541909			
2	CustomerID	406829			
3	StockCode	541909			
4	Quantity	541909			
5	UnitPrice	541909			
6	InvoiceDate	541909			
7	InvoiceNo	541909			
8	Description	540455			

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

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_project1.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_project1.data;
```

제목 없는 쿼리 실행 다운로드 공유 일정 다음에서 열기 더보기 저장 실행 시 이 쿼리가 17.87MB를 처리합니다.

```
1 SELECT
2   'CustomerID' AS column_name,
3   ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
4 FROM modulabs_project1.data
5 UNION ALL
6 SELECT
7   'Description' AS column_name,
8   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
9 FROM modulabs_project1.data;
--
```

최근성 옵션을 보려면 Option+F1 키를 누르세요.

쿼리 결과 결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	column_name	missing_percentage			
1	Description	0.27			
2	CustomerID	24.93			

결측치 처리 전략

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

```
SELECT Description, COUNT(Description)
FROM modulabs_project1.data
WHERE StockCode = '85123A'
GROUP BY Description;
```

The screenshot shows a SQL query execution interface. The query is: `SELECT Description, COUNT(Description) FROM modulabs_project1.data WHERE StockCode = '85123A' GROUP BY Description;`. The results are displayed in a table with 4 rows and 2 columns: `Description` and `row_`. The data is as follows:

row_	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_project1.data`
WHERE CustomerID IS NULL;
```

The screenshot shows a SQL query execution interface. The query is: `DELETE FROM `modulabs_project1.data` WHERE CustomerID IS NULL;`. The results are displayed in a table with 1 row and 1 column: `row_`. The data is as follows:

row_
1

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT *
FROM project_name.modulabs_project.data
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

중복값 처리

- 중복값을 제거하는 쿼리문 작성하기

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

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

[결과 이미지를 넣어주세요]

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

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

[결과 이미지를 넣어주세요]

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

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

[결과 이미지를 넣어주세요]

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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]]
LIMIT 100;
```

[결과 이미지를 넣어주세요]

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

```
SELECT ROUND(SUM(CASE WHEN # [[YOUR QUERY]] THEN 1 ELSE 0 END) / # [[YOUR QUERY]], 1)
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

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

[결과 이미지를 넣어주세요]

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

- 상위 10개의 제품들을 출력하기

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

```
ORDER BY sell_cnt DESC
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

- **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 project_name.modulabs_project.data
)
WHERE # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

- **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
2)
`user_rfm` 테이블과 결과를 합치기
3)
`user_data` 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

2. 평균 구매 주기

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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_data AS
WITH purchase_intervals AS (
  -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_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
      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]

Keep : 집중한다.

Problem : 시간 배분이 안됐다.

Try : 소요예상 시간을 예상하고 나머지 하자.