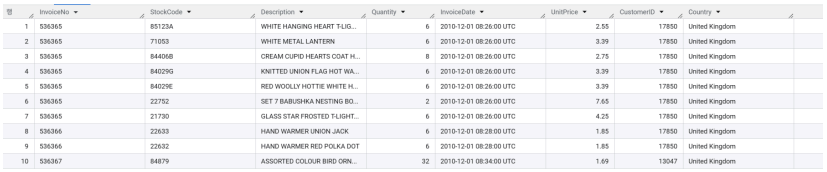
# 고객을 세그먼테이션하자 [프로젝트]

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

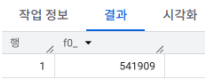
```
SELECT * FROM `modulabs_project.data` LIMIT 10
```



| 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 Kingdom |
| 2 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 2010-12-01 08:26:00 UTC | 3.39 | 17850 | United Kingdom |
| 3 | 536365 | 844068 | 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(*) FROM `modulabs_project.data`
```



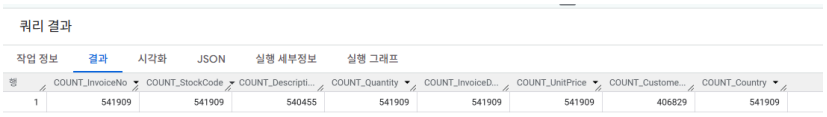
작업 정보 | 결과 | 시각화

| 행 | f0_ |
|---|---|
| 1 | 541909 |

### 데이터 수 세기

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

```
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`
```



쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

| 행 | COUNT_InvoiceNo | COUNT_StockCode | COUNT_Descripti... | COUNT_Quantity | COUNT_InvoiceD... | COUNT_UnitPrice | COUNT_Custome... | 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**

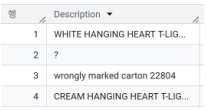
```
SELECT
   'InvoiceNo' AS missing_value,
   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `modulabs_project.data`

UNION ALL

SELECT
   'stockCode' AS missing_value,
   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 missing_value,
   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 missing_value,
   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 missing_value,
   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 missing_value,
   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 missing_value,
   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 missing_value,
   ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `modulabs_project.data`;
```

| 행 | missing_value | missing_percenta... |
|---|---|---|
| 1 | CustomerID | 24.93 |
| 2 | InvoiceNo | 0.0 |
| 3 | stockCode | 0.0 |
| 4 | InvoiceDate | 0.0 |
| 5 | Description | 0.27 |
| 6 | Quantity | 0.0 |
| 7 | UnitPrice | 0.0 |
| 8 | Country | 0.0 |

## 결측치 처리 전략

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

```
SELECT DISTINCT
Description
FROM `modulabs_project.data`
WHERE StockCode = '85123A'
```



| 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 Description IS NULL
  OR Description = '?';

DELETE FROM `modulabs_project.data`
WHERE CustomerID IS NULL
```


# 11-5. 데이터 전처리(2): 중복값 처리

## 중복값 확인

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

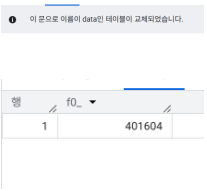
```
SELECT *
FROM modulabs_project.data
GROUP BY
   InvoiceNo,stockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING
   COUNT(*) > 1
```

| | InvoiceNo | StockCode | Description |
|---|---|---|---|
| 1 | 537534 | 22520 | CHILDS GA |
| 2 | 539733 | 22962 | JAM JAR W |
| 3 | 539856 | 22518 | CHILDS GA |
| 4 | 541975 | 20682 | RED RETRO |
| 5 | 541975 | 20749 | ASSORTED |
| 6 | 541975 | 22452 | MEASURIN |
| 7 | 542867 | 22131 | FOOD CON |
| 8 | 543899 | 21121 | SET/10 REI |
| 9 | 547224 | 22945 | CHRISTMA |
| 10 | 548604 | 22453 | MEASURIN |
| 11 | 549537 | 22939 | APRON AP |
| 12 | 549714 | 72741 | GRAND CH |
| 13 | 553521 | 22684 | FRENCH BI |
| 14 | 557644 | 23290 | SPACEBOY |

### 중복값 처리

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

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT DISTINCT
   *
FROM
   modulabs_project.data
```

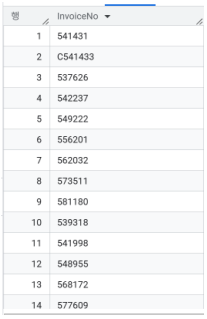



## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- **고유(unique)한** `InvoiceNo` **의 개수를 출력하기**

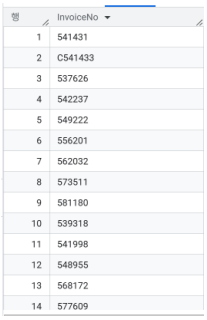
```
SELECT
   COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM
   modulabs_project.data
```



| 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 |

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

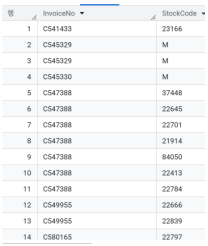
```
SELECT DISTINCT
  InvoiceNo
FROM
  modulabs_project.data
LIMIT 100
```



| 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 |

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

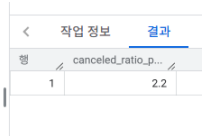
```
SELECT *
FROM project_name.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```



| 행 | InvoiceNo | StockCode |
|---|---|---|
| 1 | C541433 | 23166 |
| 2 | C545329 | M |
| 3 | C545329 | M |
| 4 | C545330 | M |
| 5 | C547388 | 37448 |
| 6 | C547388 | 22645 |
| 7 | C547388 | 22701 |
| 8 | C547388 | 21914 |
| 9 | C547388 | 84050 |
| 10 | C547388 | 22413 |
| 11 | C547388 | 22784 |
| 12 | C549955 | 22666 |
| 13 | C549955 | 22839 |
| 14 | C580165 | 22797 |

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

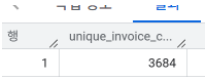
```
SELECT
  ROUND(
    (SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) * 100.0) / COUNT(*),
    1
  ) AS canceled_ratio_percent
FROM
  project_name.modulabs_project.data
```




## StockCode 살펴보기

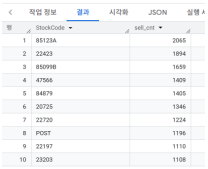
- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT
   COUNT(DISTINCT StockCode) AS unique_invoice_count
FROM
   modulabs_project.data
```




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

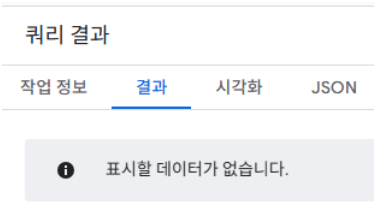
```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM project_name.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```




- **`StockCode`의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값**들에는 어떤 코드들이 들어가 있는지 출력하기

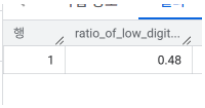
```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM modulabs_project.data
)
WHERE number_count <= 1;
```

### 실수로 빼먹어서 지워진 값 → 없는 데이터로 나와요..



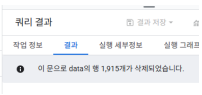

- `StockCode`**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
SELECT
  ROUND(
    (SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) * 100.0) / COUNT(*),
    2
  ) AS ratio_of_low_digit_codes
FROM
  (
    SELECT
      StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM
      modulabs_project.data
  ) AS subquery;
```




- **제품과 관련되지 않은 거래 기록을 제거하기**

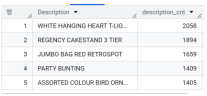
```
DELETE FROM modulabs_project.data
WHERE StockCode IN (
 SELECT DISTINCT StockCode
 FROM (
  SELECT
   StockCode,
   LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM
   modulabs_project.data
 )
 WHERE number_count <= 1
);
```




## Description 살펴보기

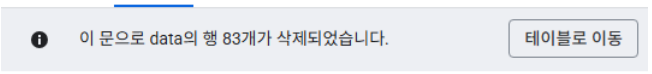
- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT
   Description,
   COUNT(*) AS description_cnt
FROM
   project_name.modulabs_project.data
GROUP BY
   Description
ORDER BY
   description_cnt DESC
LIMIT 30;
```



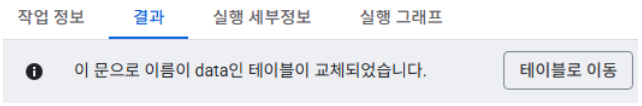
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE FROM modulabs_project.data
WHERE
   Description LIKE '%Next Day Carriage%' OR Description LIKE '%High Resolution Image%';
```




- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

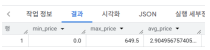
```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM modulabs_project.data
```




## `UnitPrice` 살펴보기

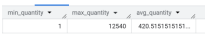
- `UnitPrice`**의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
MIN(UnitPrice) AS min_price,
MAX(UnitPrice) AS max_price,
AVG(UnitPrice) AS avg_price
FROM project_name.modulabs_project.data;
```



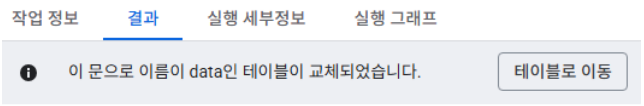
- **단가가 0원인 거래의 개수, 구매 수량( `Quantity` )의 최솟값, 최댓값, 평균 구하기**

```
SELECT
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM
  modulabs_project.data
WHERE
  UnitPrice = 0
```



- **`UnitPrice = 0` 를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT *
FROM modulabs_project.data
WHERE UnitPrice > 0;
```

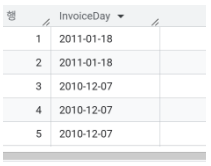



# 11-7. RFM 스코어

## Recency

- **`InvoiceDate` 컬럼을 연월일 자료형으로 변경하기**

```
SELECT Date(InvoiceDate) As InvoiceDay
FROM project_name.modulabs_project.data;
```

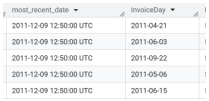


| 행 | InvoiceDay |
|---|---|
| 1 | 2011-01-18 |
| 2 | 2011-01-18 |
| 3 | 2010-12-07 |
| 4 | 2010-12-07 |
| 5 | 2010-12-07 |

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(InvoiceDate) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
```

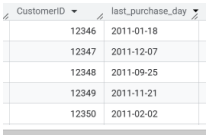
```
FROM
  modulabs_project.data;
```



| most_recent_date | InvoiceDay |
|---|---|
| 2011-12-09 12:50:00 UTC | 2011-04-21 |
| 2011-12-09 12:50:00 UTC | 2011-06-03 |
| 2011-12-09 12:50:00 UTC | 2011-09-22 |
| 2011-12-09 12:50:00 UTC | 2011-05-06 |
| 2011-12-09 12:50:00 UTC | 2011-06-15 |

- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

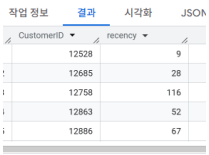
```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS last_purchase_day
FROM
  modulabs_project.data
GROUP BY
  CustomerID
```



| CustomerID | last_purchase_day |
|---|---|
| 12346 | 2011-01-18 |
| 12347 | 2011-12-07 |
| 12348 | 2011-09-25 |
| 12349 | 2011-11-21 |
| 12350 | 2011-02-02 |

- **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
);
```



작업 정보 결과 시각화 JSON

| CustomerID | recency |
|---|---|
| 12528 | 9 |
| 12685 | 28 |
| 12758 | 116 |
| 12863 | 52 |
| 12886 | 67 |

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

```
CREATE OR REPLACE TABLE modulabs_project.user_r AS
SELECT
  CustomerID,
  DATE_DIFF(
    (SELECT MAX(DATE(InvoiceDate)) FROM modulabs_project.data),
    MAX(DATE(InvoiceDate)),
    DAY
  ) AS recency
FROM
  modulabs_project.data
```

```
WHERE
  CustomerID IS NOT NULL
GROUP BY
  CustomerID;
```

| 행 | CustomerID | recency |
|---|---|---|
| 1 | 13113 | 0 |
| 2 | 16705 | 0 |
| 3 | 12518 | 0 |
| 4 | 14446 | 0 |
| 5 | 16954 | 0 |
| 6 | 12433 | 0 |
| 7 | 12423 | 0 |
| 8 | 14422 | 0 |
| 9 | 12662 | 0 |
| 10 | 15344 | 0 |
| 11 | 17364 | 0 |
| 12 | 18102 | 0 |
| 13 | 12985 | 0 |

## Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

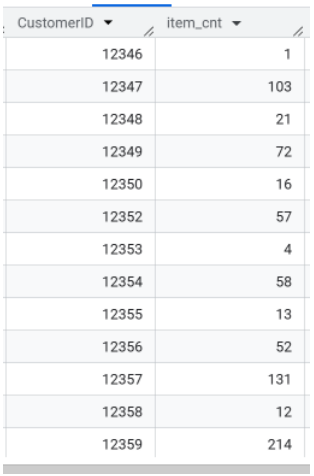
```
SELECT
  CustomerID,
  COUNT(InvoiceNO) AS purchase_cnt
FROM project_name.modulabs_project.data
GROUP BY CustomerID;
```

| | 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 |
| 0 | 12356 | 3 |
| 1 | 12357 | 1 |
| 2 | 12358 | 2 |
| 3 | 12359 | 6 |
| 4 | 12360 | 3 |

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM
  modulabs_project.data
WHERE CustomerID IS NOT NULL
```

```
GROUP BY
    CustomerID;
```



| CustomerID | item_cnt |
|---|---|
| 12346 | 1 |
| 12347 | 103 |
| 12348 | 21 |
| 12349 | 72 |
| 12350 | 16 |
| 12352 | 57 |
| 12353 | 4 |
| 12354 | 58 |
| 12355 | 13 |
| 12356 | 52 |
| 12357 | 131 |
| 12358 | 12 |
| 12359 | 214 |

- **전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 `user_rf` 라는 이름의 테이블에 저장하기**

```
CREATE OR REPLACE TABLE modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산 (Frequency: 고유 송장 번호 수)
WITH purchase_cnt AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT InvoiceNo) AS purchase_cnt
    FROM
        modulabs_project.data
    WHERE
        CustomerID IS NOT NULL
    GROUP BY
        CustomerID
),

-- (2) 구매한 아이템 총 수량 계산 (Volume: Quantity 합계)
item_cnt AS (
    SELECT
        CustomerID,
        SUM(Quantity) AS item_cnt
    FROM
        modulabs_project.data
    WHERE
        CustomerID IS NOT NULL
    GROUP BY
        CustomerID
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
    pc.CustomerID,
    pc.purchase_cnt,
    ic.item_cnt,
    ur.recency
FROM
    purchase_cnt AS pc
INNER JOIN
    item_cnt AS ic
```

```
    ON pc.CustomerID = ic.CustomerID
INNER JOIN
    modulabs_project.user_r AS ur
    ON pc.CustomerID = ur.CustomerID;
```

| 행 | CustomerID | purchase_cnt | item_cnt | recency |
|---|---|---|---|---|
| 1 | 12713 | 1 | 505 | 0 |
| 2 | 15520 | 1 | 314 | 1 |
| 3 | 13298 | 1 | 96 | 1 |
| 4 | 14569 | 1 | 79 | 1 |
| 5 | 13436 | 1 | 76 | 1 |
| 6 | 14204 | 1 | 72 | 2 |
| 7 | 15471 | 1 | 256 | 2 |
| 8 | 15195 | 1 | 1404 | 2 |
| 9 | 17914 | 1 | 457 | 3 |
| 10 | 16528 | 1 | 171 | 3 |
| 11 | 12478 | 1 | 233 | 3 |

## Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
FROM
    modulabs_project.data
WHERE
    CustomerID IS NOT NULL
GROUP BY
    CustomerID;
```

| | CustomerID | user_total |
|---|---|---|
| 1 | 12346 | 0.0 |
| 2 | 12347 | 4310.0 |
| 3 | 12348 | 1437.0 |
| 4 | 12349 | 1458.0 |
| 5 | 12350 | 294.0 |
| 6 | 12352 | 1265.0 |
| 7 | 12353 | 89.0 |
| 8 | 12354 | 1079.0 |
| 9 | 12355 | 459.0 |

- **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 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) AS user_average
```

```
FROM
  modulabs_project.user_rf rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM
    modulabs_project.data
  WHERE
    CustomerID IS NOT NULL
  GROUP BY
    CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

### RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select * from `modulabs_project.user_rfm`
```

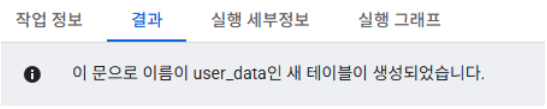
| | CustomerID | purchase_cnt | item_cnt | recency | user_total | user_average |
|---|---|---|---|---|---|---|
| 1 | 12713 | 1 | 505 | 0 | 795.0 | 795.0 |
| 2 | 13298 | 1 | 96 | 1 | 360.0 | 360.0 |
| 3 | 13436 | 1 | 76 | 1 | 197.0 | 197.0 |
| 4 | 14569 | 1 | 79 | 1 | 227.0 | 227.0 |
| 5 | 15520 | 1 | 314 | 1 | 343.0 | 343.0 |
| 6 | 14204 | 1 | 72 | 2 | 151.0 | 151.0 |
| 7 | 15471 | 1 | 256 | 2 | 454.0 | 454.0 |
| 8 | 15195 | 1 | 1404 | 2 | 3861.0 | 3861.0 |

## 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**

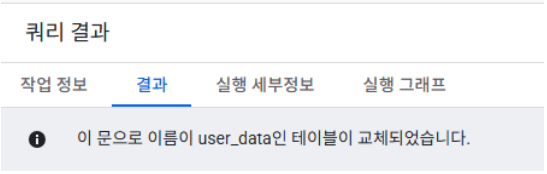
```
CREATE OR REPLACE TABLE modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM modulabs_project.user_rfm AS ur
JOIN unique_products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




## 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;
```

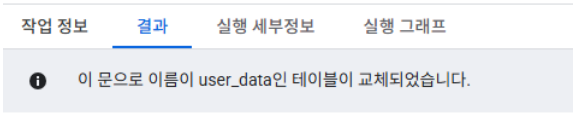



## 3. 구매 취소 경향성

- **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 modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
```

```
  FROM
    modulabs_project.data
  WHERE
    CustomerID IS NOT NULL
  GROUP BY
    CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  ROUND((t.cancel_frequency * 100.0) / t.total_transactions, 2) AS cancel_rate
FROM
  modulabs_project.user_data AS u
LEFT JOIN
  TransactionInfo AS t
ON
  u.CustomerID = t.CustomerID;
```




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

```
select * from modulabs_project.user_data
```

| 행 | CustomerID | purchase_cnt | item_cnt | recency | user_total | user_average | un |
|---|---|---|---|---|---|---|---|
| 12 | 17731 | 5 | 392 | 79 | 878.0 | 176.0 | |
| 13 | 14389 | 5 | 702 | 8 | 1000.0 | 200.0 | |
| 14 | 13680 | 5 | 1265 | 1 | 2199.0 | 440.0 | |
| 15 | 16801 | 5 | 307 | 163 | 842.0 | 168.0 | |
| 16 | 15786 | 5 | 3307 | 43 | 4821.0 | 964.0 | |
| 17 | 16571 | 5 | 1176 | 3 | 1285.0 | 257.0 | |
| 18 | 16859 | 5 | 655 | 106 | 458.0 | 92.0 | |
| 19 | 15676 | 5 | 2230 | 9 | 3309.0 | 662.0 | |

# 회고

[회고 내용을 작성해주세요]

Keep : 오늘 문제가 발생해도 계속 원인을 찾으려고 하고 결국에는 문제를 잘 수정했다는 것! 앞으로도 잘 했으면 좋겠다. 이전시간에 배운 내용들을 많이 기억해서 잘 사용하려고 해서 잘 문제를 풀어낼 수 있었다. (아닌 것들도 있었음) 그치만 이전 내용들 컨닝하면서 많이 쓰려고 한 점 굿굿. 데이터 정제 부분은 정말 쉽게 넘어와서 다행이다.

Problem : 중간에 frequency 부분에서 오류가 나고 멘붕이 와서 monetary에서 완전 바보같은 행동을 한 점은 반성해야한다. 제정신이 아닌 채로 쿼리를 짜니깐 실수가 눈에 들어오지 않고 다른 곳으로 신경이 쏠려서 제대로된 문제점을 파악할 수 없었다. 정신을 차리고 다시 보니깐 눈에 들어오긴 했지만 (퍼실님의 한마디에..) 앞으로는 이런 일이 없었으면 좋겠다. 시간을 너무 잡아먹었다.

Try : 어제 조퇴를 해서 어제 노드학습을 제대로 하지 않은 상태에서 오늘 노드를 하기 시작했다. 어제 노드를 완벽하게 숙지하고 오늘 것을 했으면 더 수월하게 진행했을 것 같다. 앞으로는 전 노드를 잘 확인하고 예습복습 철저히!!!하자..(은별님이 복습을 아주 열심히하더군요..)