# 고객을 세그먼테이션하자 [프로젝트] - 김효주 (1)

## 11-2. 데이터 불러오기

### 데이터 살펴보기

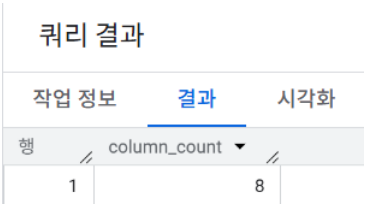
- **테이블에 있는 10개의 행만 출력하기**

```
# SELECT *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
LIMIT 10;
```

| 행 | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice |
|---|---|---|---|---|---|---|
| 1 | 536365 | 85123A | WHITE HANGING HEART T-LIG... | 6 | 2010-12-01 08:26:00 UTC | 2.5 |
| 2 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 2010-12-01 08:26:00 UTC | 3.3 |
| 3 | 536365 | 84406B | CREAM CUPID HEARTS COAT H... | 8 | 2010-12-01 08:26:00 UTC | 2.7 |
| 4 | 536365 | 84029G | KNITTED UNION FLAG HOT WA... | 6 | 2010-12-01 08:26:00 UTC | 3.3 |
| 5 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE H... | 6 | 2010-12-01 08:26:00 UTC | 3.3 |

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*) AS column_count
FROM `project-9f416757-b7a0-4248-903.modulabs_project.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'data';
```




### 데이터 수 세기

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

```
--각 컬럼별 데이터 포인트의 수
SELECT
  COUNT(InvoiceNo) AS InvoiceNo_count,
  COUNT(StockCode) AS StockCode_count,
  COUNT(Description) AS Description_count,
  COUNT(Quantity) AS Quantity_count,
  COUNT(InvoiceDate) AS InvoiceDate_count,
  COUNT(UnitPrice) AS UnitPrice_count,
  COUNT(CustomerID) AS CustomerID_count,
  COUNT(Country) AS Country_count
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```

| 행 | InvoiceNo_count | StockCode_count | Description_count | Quantity_count | InvoiceDate_count | UnitPrice_count | CustomerID_count | Country_count |
|---|---|---|---|---|---|---|---|---|
| 1 | 541909 | 541909 | 540455 | 541909 | 541909 | 541909 | 406829 | 54 |

## 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**

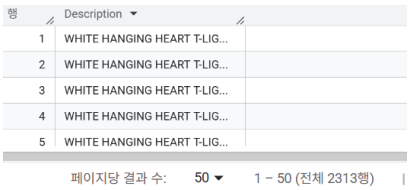
```
SELECT
  'InvoiceNo' AS column_name,
  COUNTIF(InvoiceNo IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`
UNION ALL
SELECT
  'StockCode' AS column_name,
  COUNTIF(StockCode IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`
UNION ALL
SELECT
  'Description' AS column_name,
  COUNTIF(Description IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`
UNION ALL
SELECT
  'Quantity' AS column_name,
  COUNTIF(Quantity IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`
UNION ALL
SELECT
  'InvoiceDate' AS column_name,
  COUNTIF(InvoiceDate IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`
UNION ALL
SELECT
  'UnitPrice' AS column_name,
  COUNTIF(UnitPrice IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`
UNION ALL
SELECT
  'CustomerID' AS column_name,
  COUNTIF(CustomerID IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`
UNION ALL
SELECT
  'Country' AS column_name,
  COUNTIF(Country IS NULL) AS missing_count
FROM
  `project-9f416757-b7a0-4248-903`.`modulabs_project`.`data`;
```

| 행 | column_name | missing_count |
|---|---|---|
| 1 | InvoiceNo | 0 |
| 2 | StockCode | 0 |
| 3 | Description | 1454 |
| 4 | Quantity | 0 |
| 5 | InvoiceDate | 0 |
| 6 | UnitPrice | 0 |
| 7 | CustomerID | 135080 |
| 8 | Country | 0 |

## 결측치 처리 전략

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

```
SELECT Description
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE StockCode = '85123A';
```



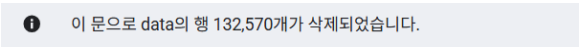
| 행 | Description |
|---|---|
| 1 | WHITE HANGING HEART T-LIG... |
| 2 | WHITE HANGING HEART T-LIG... |
| 3 | WHITE HANGING HEART T-LIG... |
| 4 | WHITE HANGING HEART T-LIG... |
| 5 | WHITE HANGING HEART T-LIG... |

페이지당 결과 수: 50 1 – 50 (전체 2313행)

## 결측치 처리

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

```
DELETE FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE Description IS NULL
  OR CustomerID IS NULL;
```

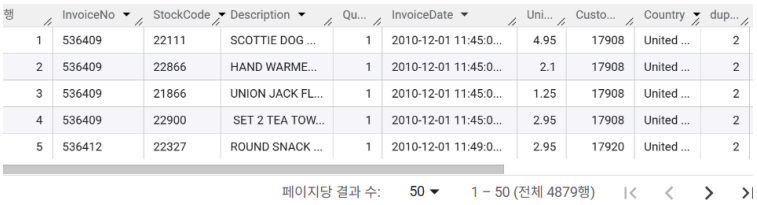



# 11-5. 데이터 전처리(2): 중복값 처리

## 중복값 확인

- **중복된 행의 수를 세어보기**
- SELECT * → SELECT 컬럼명 전부 : GROUP BY 시 SELECT에 있는 컬럼을 모두 명시
- dup_cnt 추가 : 중복된 행이 몇 번 등장했는지 알려줌
  - **8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기**

```
SELECT
  InvoiceNo,
  StockCode,
  Description,
  Quantity,
  InvoiceDate,
  UnitPrice,
  CustomerID,
  Country,
  COUNT(*) AS dup_cnt
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
GROUP BY
  InvoiceNo,
  StockCode,
  Description,
  Quantity,
  InvoiceDate,
  UnitPrice,
  CustomerID,
  Country
HAVING COUNT(*) > 1;
```



| 행 | InvoiceNo | StockCode | Description | Qu... | InvoiceDate | Uni... | Custo... | Country | dup... |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 536409 | 22111 | SCOTTIE DOG ... | 1 | 2010-12-01 11:45:0... | 4.95 | 17908 | United ... | 2 |
| 2 | 536409 | 22866 | HAND WARME... | 1 | 2010-12-01 11:45:0... | 2.1 | 17908 | United ... | 2 |
| 3 | 536409 | 21866 | UNION JACK FL... | 1 | 2010-12-01 11:45:0... | 1.25 | 17908 | United ... | 2 |
| 4 | 536409 | 22900 | SET 2 TEA TOW... | 1 | 2010-12-01 11:45:0... | 2.95 | 17908 | United ... | 2 |
| 5 | 536412 | 22327 | ROUND SNACK ... | 1 | 2010-12-01 11:49:0... | 2.95 | 17920 | United ... | 2 |

페이지당 결과 수: 50 1 – 50 (전체 4879행)

### 중복값 처리

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

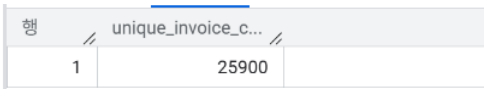
```
CREATE OR REPLACE TABLE `project-9f416757-b7a0-4248-903.modulabs_project.data` AS
SELECT DISTINCT *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```

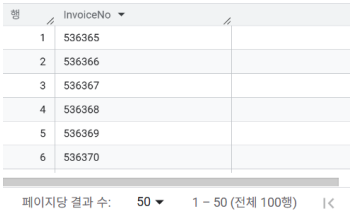


| 행 | unique_invoice_c... |
|---|---|
| 1 | 25900 |

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
ORDER BY InvoiceNo
LIMIT 100;
```

[결과 이미지를 넣어주세요]



| 행 | InvoiceNo |
|---|---|
| 1 | 536365 |
| 2 | 536366 |
| 3 | 536367 |
| 4 | 536368 |
| 5 | 536369 |
| 6 | 536370 |

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

| 행 | Invoice... | Stoc... | Description | Qua... | InvoiceDate | Uni... | Cust... | Country |
|---|---|---|---|---|---|---|---|---|
| 1 | C537251 | 21891 | TRADITIONAL ... | -3 | 2010-12-06 10:4... | 1.25 | null | United Kingdom |
| 2 | C537251 | 22466 | FAIRY TALE CO... | -3 | 2010-12-06 10:4... | 1.95 | null | United Kingdom |
| 3 | C537251 | 22911 | PAPER CHAIN ... | -2 | 2010-12-06 10:4... | 2.95 | null | United Kingdom |
| 4 | C537251 | 21826 | EIGHT PIECE DI... | -4 | 2010-12-06 10:4... | 1.25 | null | United Kingdom |
| 5 | C537251 | 22418 | 10 COLOUR SP... | -7 | 2010-12-06 10:4... | 0.85 | null | United Kingdom |

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
--InvoiceNo LIKE 'C%' =canceled
--Canceled 인 데이터의 비율(%)=canceled_ratio_pct

SELECT ROUND(SUM(CASE WHEN  InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/ COUNT(*) * 100,
       1) AS canceled_ratio_pct
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```

| 행 | canceled_ratio_pct |
|---|---|
| 1 | 1.7 |

## `StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```

| 행 | unique_stockcod... |
|---|---|
| 1 | 3676 |

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

| 행 | StockCode | sell_cnt |
|---|---|---|
| 1 | 85123A | 2065 |
| 2 | 22423 | 1894 |
| 3 | 85099B | 1659 |
| 4 | 47566 | 1409 |
| 5 | 84879 | 1405 |

페이지당 결과 수: 50 1 – 10 (전체 10행)

- **`StockCode`의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**

`LENGTH(StockCode) - LENGTH(숫자 제거) =StockCode 안의 숫자 개수`

  - **숫자가 0~1개인 값**들에는 어떤 코드들이 들어가 있는지 출력하기

```
--숫자가 0~1개인 값=BETWEEN 0 AND 1
SELECT DISTINCT StockCode, number_count
```

```
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
)
WHERE number_count BETWEEN 0 AND 1;
```

| 행 | number_count | stock_cnt |
|---|---|---|
| 1 | 5 | 3676 |

- `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 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 number_count BETWEEN 0 AND 1;
```

| 행 | StockCode | number_count |
|---|---|---|
| 1 | DOT | 0 |
| 2 | M | 0 |
| 3 | AMAZONFEE | 0 |
| 4 | C2 | 1 |
| 5 | m | 0 |

페이지당 결과 수: 50 1 – 14 (전체 14행)

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `project-9f416757-b7a0-4248-903.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 `project-9f416757-b7a0-4248-903.modulabs_project.data`
  )
  WHERE number_count BETWEEN 0 AND 1
  );
```

ⓘ 이 문으로 data의 행 2,934개가 삭제되었습니다.

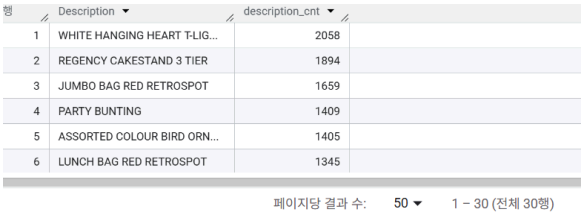
## Description 살펴보기

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

```
SELECT
  Description,
```

```
  COUNT(*) AS description_cnt
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]



| 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 |

페이지당 결과 수: 50 1 – 30 (전체 30행)

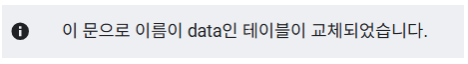
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE UPPER(Description) LIKE '%POSTAGE%'
  OR UPPER(Description) LIKE '%CARRIAGE%'
  OR UPPER(Description) LIKE '%SAMPLE%'
  OR UPPER(Description) LIKE '%BANK CHARGES%';
```

[결과 이미지를 넣어주세요]


- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

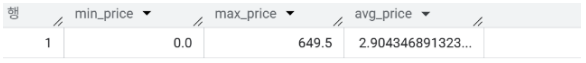
```
CREATE OR REPLACE TABLE `project-9f416757-b7a0-4248-903.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```




## UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```



| 행 | min_price | max_price | avg_price |
|---|---|---|---|
| 1 | 0.0 | 649.5 | 2.904346891323... |

- **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,
  AVG(Quantity) AS avg_quantity
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE UnitPrice = 0;
```

| 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 `project-9f416757-b7a0-4248-903.modulabs_project.data` AS
SELECT *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE UnitPrice > 0;
```

ⓘ 이 문으로 이름이 data인 테이블이 교체되었습니다.

# 11-7. RFM 스코어

## Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```

| 행 | InvoiceDay | InvoiceNo | StockCode | Quantity | InvoiceDate | UnitPrice |
|---|---|---|---|---|---|---|
| 1 | 2011-01-18 | 541431 | 23166 | 74215 | 2011-01-18 10:01:00 UTC | 1.04 |
| 2 | 2011-01-18 | C541433 | 23166 | -74215 | 2011-01-18 10:17:00 UTC | 1.04 |
| 3 | 2010-12-07 | 537626 | 84969 | 6 | 2010-12-07 14:57:00 UTC | 4.25 |
| 4 | 2010-12-07 | 537626 | 85116 | 12 | 2010-12-07 14:57:00 UTC | 2.1 |
| 5 | 2010-12-07 | 537626 | 84558A | 24 | 2010-12-07 14:57:00 UTC | 2.95 |
| 6 | 2010-12-07 | 537626 | 22195 | 12 | 2010-12-07 14:57:00 UTC | 1.65 |

페이지당 결과 수: 50 1 – 50 (전체 399510행)

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

| 행 | most_recent_date | InvoiceDay | InvoiceNo | StockCode | Quantity | InvoiceDate |
|---|---|---|---|---|---|---|
| 1 | 2011-12-09 | 2011-01-18 | 541431 | 23166 | 74215 | 2011-01-18 10:01:00 UTC |
| 2 | 2011-12-09 | 2011-01-18 | C541433 | 23166 | -74215 | 2011-01-18 10:17:00 UTC |
| 3 | 2011-12-09 | 2010-12-07 | 537626 | 84969 | 6 | 2010-12-07 14:57:00 UTC |
| 4 | 2011-12-09 | 2010-12-07 | 537626 | 85116 | 12 | 2010-12-07 14:57:00 UTC |
| 5 | 2011-12-09 | 2010-12-07 | 537626 | 84558A | 24 | 2010-12-07 14:57:00 UTC |
| 6 | 2011-12-09 | 2010-12-07 | 537626 | 22195 | 12 | 2010-12-07 14:57:00 UTC |

페이지당 결과 수: 50 1 – 50 (전체 399510행)

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

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

| 행 | CustomerID | InvoiceDay |
|---|---|---|
| 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 |

- **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-9f416757-b7a0-4248-903.modulabs_project.data`
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

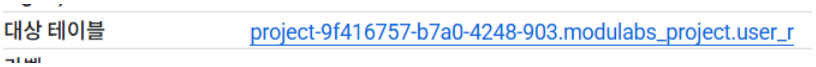
| 행 | CustomerID | recency |
|---|---|---|
| 1 | 12381 | 4 |
| 2 | 12401 | 303 |
| 3 | 12674 | 11 |
| 4 | 12766 | 3 |
| 5 | 12944 | 35 |
| 6 | 13121 | 269 |

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

```
CREATE OR REPLACE TABLE `project-9f416757-b7a0-4248-903.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 `project-9f416757-b7a0-4248-903.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]




## Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

| 행 | CustomerID | purchase_cnt |
|---|---|---|
| 1 | 12346 | 2 |
| 2 | 12347 | 7 |
| 3 | 12348 | 4 |
| 4 | 12349 | 1 |
| 5 | 12350 | 1 |

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

| 행 | CustomerID | item_cnt |
|---|---|---|
| 1 | 12346 | 0 |
| 2 | 12347 | 2458 |
| 3 | 12348 | 2332 |
| 4 | 12349 | 630 |
| 5 | 12350 | 196 |

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

```
CREATE OR REPLACE TABLE `project-9f416757-b7a0-4248-903.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `project-9f416757-b7a0-4248-903.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 `project-9f416757-b7a0-4248-903.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
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN `project-9f416757-b7a0-4248-903.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

| 행 | CustomerID | purchase_cnt | item_cnt | recency |
|---|---|---|---|---|
| 1 | 12713 | 1 | 505 | 0 |
| 2 | 13298 | 1 | 96 | 1 |
| 3 | 13436 | 1 | 76 | 1 |
| 4 | 15520 | 1 | 314 | 1 |
| 5 | 14569 | 1 | 79 | 1 |

## Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

| 행 | CustomerID | user_total |
|---|---|---|
| 1 | 12346 | 0.0 |
| 2 | 12347 | 4310.0 |
| 3 | 12348 | 1437.2 |
| 4 | 12349 | 1457.5 |
| 5 | 12350 | 294.4 |

- **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-9f416757-b7a0-4248-903.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 `project-9f416757-b7a0-4248-903.modulabs_project.user_rf` rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

| 행 | CustomerID | purchase_cnt | item_cnt | recency | user_total | user_average |
|---|---|---|---|---|---|---|
| 1 | 12713 | 1 | 505 | 0 | 794.6 | 794.6 |
| 2 | 14569 | 1 | 79 | 1 | 227.4 | 227.4 |
| 3 | 13298 | 1 | 96 | 1 | 360.0 | 360.0 |
| 4 | 13436 | 1 | 76 | 1 | 196.9 | 196.9 |
| 5 | 15520 | 1 | 314 | 1 | 343.5 | 343.5 |

## RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.user_rfm`;
```

| 행 | CustomerID | purchase_cnt | item_cnt | recency | user_total | user_average |
|---|---|---|---|---|---|---|
| 1 | 12713 | 1 | 505 | 0 | 794.6 | 794.6 |
| 2 | 14569 | 1 | 79 | 1 | 227.4 | 227.4 |
| 3 | 13298 | 1 | 96 | 1 | 360.0 | 360.0 |
| 4 | 13436 | 1 | 76 | 1 | 196.9 | 196.9 |
| 5 | 15520 | 1 | 314 | 1 | 343.5 | 343.5 |

# 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**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID;
```

| 행 | CustomerID | user_total |
|---|---|---|
| 1 | 12346 | 0.0 |
| 2 | 12347 | 4310.0 |
| 3 | 12348 | 1437.2 |
| 4 | 12349 | 1457.5 |
| 5 | 12350 | 294.4 |

## 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-9f416757-b7a0-4248-903.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 `project-9f416757-b7a0-4248-903.modulabs_project.user_rf` rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

| 행 | CustomerID | purchase_cnt | item_cnt | recency | user_total | user_average |
|---|---|---|---|---|---|---|
| 1 | 12713 | 1 | 505 | 0 | 794.6 | 794.6 |
| 2 | 14569 | 1 | 79 | 1 | 227.4 | 227.4 |
| 3 | 13298 | 1 | 96 | 1 | 360.0 | 360.0 |
| 4 | 13436 | 1 | 76 | 1 | 196.9 | 196.9 |
| 5 | 15520 | 1 | 314 | 1 | 343.5 | 343.5 |

## 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)**

```
--SAFE_DIVIDE는 0으로 나누는 에러 방지용 (혹시 total_transactions=0인 경우를 대비해서)
CREATE OR REPLACE TABLE `project-9f416757-b7a0-4248-903.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `project-9f416757-b7a0-4248-903.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `project-9f416757-b7a0-4248-903.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

| 행 | CustomerID | purchase_cnt | item_cnt | recency | user_total | user_average | unique_prod... |
|---|---|---|---|---|---|---|---|
| 1 | 17347 | 1 | 216 | 86 | 229.0 | 229.0 | 1 |
| 2 | 13307 | 1 | 4 | 120 | 15.0 | 15.0 | 1 |
| 3 | 15389 | 1 | 400 | 172 | 500.0 | 500.0 | 1 |
| 4 | 15753 | 1 | 144 | 304 | 79.2 | 79.2 | 1 |
| 5 | 13747 | 1 | 8 | 373 | 79.6 | 79.6 | 1 |
| 6 | 12791 | 1 | 96 | 373 | 177.6 | 177.6 | 1 |
| 7 | 17763 | 1 | 12 | 263 | 15.0 | 15.0 | 1 |
| 8 | 16738 | 1 | 3 | 297 | 3.8 | 3.8 | 1 |
| 9 | 16144 | 1 | 16 | 246 | 175.2 | 175.2 | 1 |
| 10 | 13017 | 1 | 48 | 7 | 204.0 | 204.0 | 1 |

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

```
SELECT *
FROM `project-9f416757-b7a0-4248-903.modulabs_project.user_data`;
```

# 회고

[회고 내용을 작성해주세요]

Keep : 과정을 예시를 대입해서 풀어가며 이해할 수 있어서 좋았다

Problem : 익숙하지 않아 오류가 반복적으로 나와서 어려웠다.
하지만 반복적으로 수정하면서 구조와 과정에 대해 흥미가 생겼다

Try : what을 고민하고 how를 반복적으로 해결하다 보면 why가 풀릴것이다.
정리가 필요 했는데 이 과정을 통해 정리를 한것 같다.
정리한 내용을 잘 사용 할 수 있도록 나의 방법으로 수정해봐야겠다.