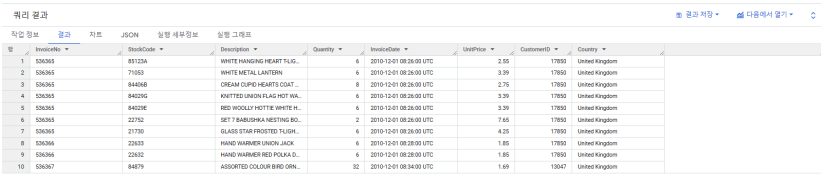
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
select * from white-might-456101-t9.modulabs_project.data limit 10;
```



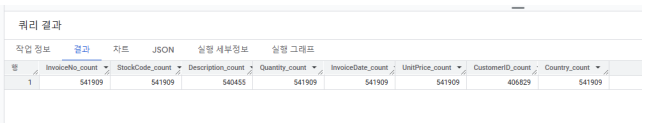
- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
select count(*) as total_count from white-might-456101-t9.modulabs_project.data;
```

데이터 수 세기

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

```
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
  `white-might-456101-t9.modulabs_project.data`;
```

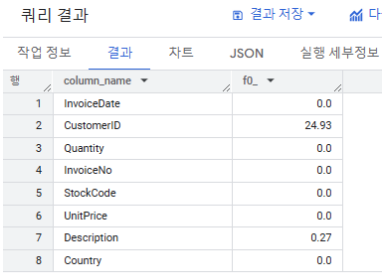


11-4. 데이터 전처리 방법(1): 결측치 제거

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 column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(
   SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM white-might-456101-t9.mo
   SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM white-might-456101-t9.m
   SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM white-might-456101-t9.
   SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM white-might-456101-t9.modul
   SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM white-might-456101-t9
   SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM white-might-456101-t9.mod
   SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM white-might-456101-t9
   SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM white-might-456101-t9.modula
) AS column_data;
```



쿼리 결과

결과 저장 ▾ 다

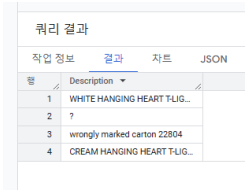
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

행	column_name	f0_
1	InvoiceDate	0.0
2	CustomerID	24.93
3	Quantity	0.0
4	InvoiceNo	0.0
5	StockCode	0.0
6	UnitPrice	0.0
7	Description	0.27
8	Country	0.0

결측치 처리 전략

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

```
select distinct Description from white-might-456101-t9.modulabs_project.data where StockCode  = '85123A';
```



쿼리 결과

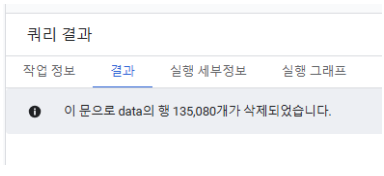
작업 정보 | 결과 | 차트 | JSON

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 `white-might-456101-t9.modulabs_project.data`
WHERE CustomerID IS NULL OR Description IS NULL;
```



쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 data의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

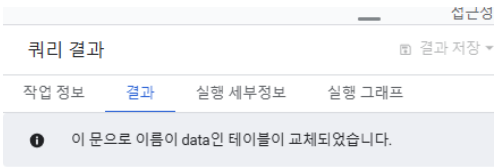
- **중복된 행의 수를 세어보기**
  - **8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기**

```
SELECT COUNT(*) AS double_count
FROM (
  SELECT *
  FROM `white-might-456101-t9.modulabs_project.data`
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
);
```

중복값 처리

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

```
CREATE OR REPLACE TABLE `white-might-456101-t9.modulabs_project.data` AS
SELECT DISTINCT *
FROM `white-might-456101-t9.modulabs_project.data`;
```

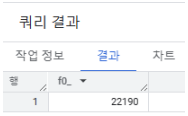


11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

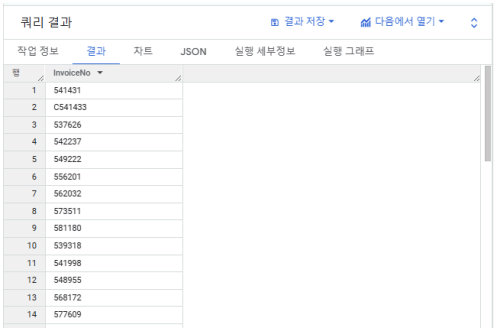
- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
select count(distinct InvoiceNo)
from white-might-456101-t9.modulabs_project.data;
```



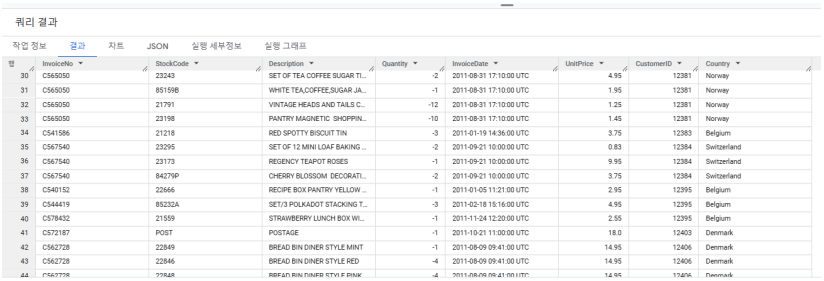
- **고유한 `InvoiceNo` 를 앞에서부터 100개를 출력하기**

```
select distinct InvoiceNo
from white-might-456101-t9.modulabs_project.data
limit 100;
```



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

```
select *
from white-might-456101-t9.modulabs_project.data
where InvoiceNo LIKE 'C%'
limit 100;
```



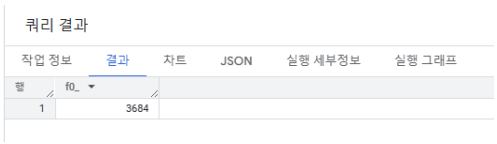
- **구매 건 상태가 `Canceled` 인 데이터의 비율(%) - 소수점 첫번째 자리까지**

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo like 'C%' THEN 1 ELSE 0 END)/ count(InvoiceNo) * 100, 1) as Canceled_Percent
from white-might-456101-t9.modulabs_project.data
```

StockCode 살펴보기

- **고유한 StockCode의 개수를 출력하기**

```
select count(distinct StockCode)
from white-might-456101-t9.modulabs_project.data;
```

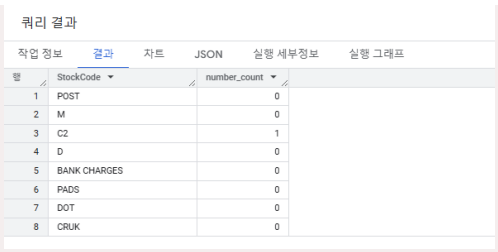


- **어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기**
  - 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
from white-might-456101-t9.modulabs_project.data
group by StockCode
ORDER BY sell_cnt DESC
limit 10;
```

- **StockCode의 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 white-might-456101-t9.modulabs_project.data
)
WHERE number_count between 0 and 1;
```



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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
-- 숫자가 0~1개인 값
WITH no_char_codes AS (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `white-might-456101-t9.modulabs_project.data`
  )
  WHERE number_count BETWEEN 0 AND 1
),

-- no_char_codes에서 distinct StockCode에 출력된 결과만을 포함하는 전체 데이터수
filter_no_char_codes AS (
select count(*) as row_count
from `white-might-456101-t9.modulabs_project.data`
where StockCode in (select StockCode from no_char_codes)
),

-- data 테이블의 전체 덴이터 행 수
total AS (
  select count(*) as total_count from `white-might-456101-t9.modulabs_project.data`
)

select ROUND( row_count / total.total_count * 100, 2) as no_char_stockcode_percent
from filter_no_char_codes, total;
```

행	no_char_stockcode_p
1	0.48

정답코드

```
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
```

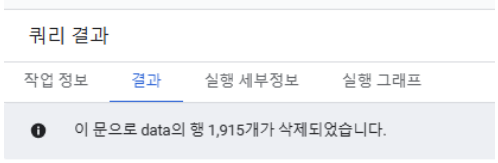
```
)
WHERE number_count < 5;
```

해당 코드 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트일까요?

```
SELECT ROUND(SUM(CASE WHEN number_count < 5 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM project_name.modulabs_project.data
);
```

- **제품과 관련되지 않은 거래 기록을 제거하기**

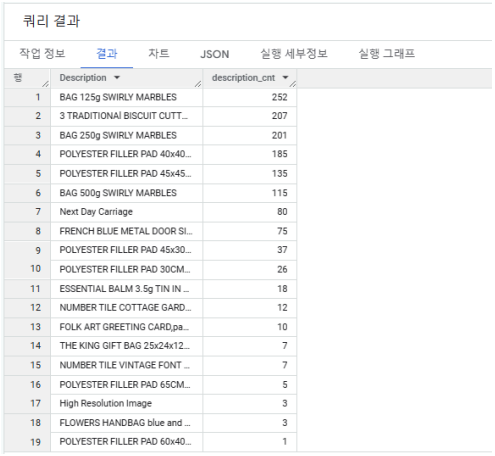
```
delete from `white-might-456101-t9.modulabs_project.data`
where StockCode in (
    SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `white-might-456101-t9.modulabs_project.data`
  )
  WHERE number_count BETWEEN 0 AND 1
);
```



Description 살펴보기

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

```
SELECT DISTINCT Description,COUNT(*) AS description_cnt
FROM `white-might-456101-t9.modulabs_project.data`
WHERE REGEXP_CONTAINS(Description, r'[a-z]')
group by Description
order by description_cnt DESC
limit 30;
```



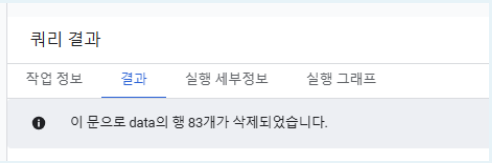
행	Description	description_cnt
1	BAG 125g SWIRLY MARBLES	252
2	3 TRADITIONAl BISCUIT CUTT...	207
3	BAG 250g SWIRLY MARBLES	201
4	POLYESTER FILLER PAD 40x40...	185
5	POLYESTER FILLER PAD 45x45...	135
6	BAG 500g SWIRLY MARBLES	115
7	Next Day Carriage	80
8	FRENCH BLUE METAL DOOR SI...	75
9	POLYESTER FILLER PAD 45x30...	37
10	POLYESTER FILLER PAD 30CM...	26
11	ESSENTIAL BALM 3.5g TIN IN ...	18
12	NUMBER TILE COTTAGE GARD...	12
13	FOLK ART GREETING CARD,pa...	10
14	THE KING GIFT BAG 25x24x12...	7
15	NUMBER TILE VINTAGE FONT ...	7
16	POLYESTER FILLER PAD 65CM...	5
17	High Resolution Image	3
18	FLOWERS HANDBAG blue and ...	3
19	POLYESTER FILLER PAD 60x40...	1

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
delete from `white-might-456101-t9.modulabs_project.data`
where Description = 'Next Day Carriage' or Description = 'High Resolution Image';
```

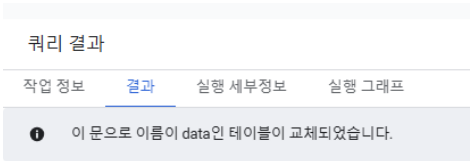
정답코드

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



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

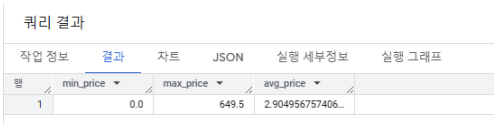
```
CREATE OR REPLACE TABLE white-might-456101-t9.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
UPPER(Description) AS Description
FROM white-might-456101-t9.modulabs_project.data
```



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
select min(UnitPrice) as min_price, max(UnitPrice) as max_price, avg(UnitPrice) as avg_price
from `white-might-456101-t9.modulabs_project.data`
```



- **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_quar
FROM `white-might-456101-t9.modulabs_project.data`
WHERE UnitPrice = 0
```

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

```
CREATE OR REPLACE TABLE `white-might-456101-t9.modulabs_project.data` AS
SELECT *
FROM `white-might-456101-t9.modulabs_project.data`
WHERE UnitPrice != 0;
```

정답 코드

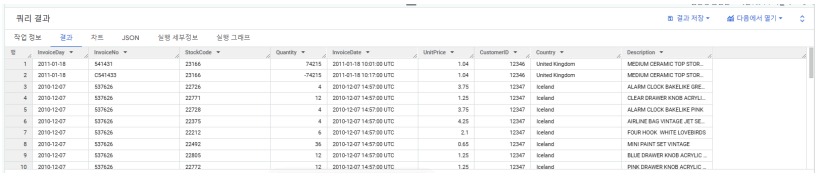
```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT *
FROM project_name.modulabs_project.data
WHERE UnitPrice > 0;
```

11-7. RFM 스코어

Recency

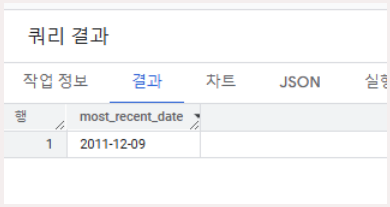
- `InvoiceDate` **컬럼을 연월일 자료형으로 변경하기**

```
SELECT Date(InvoiceDate) AS InvoiceDay, *
FROM `white-might-456101-t9.modulabs_project.data`;
```



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
FROM `white-might-456101-t9.modulabs_project.data`;
```



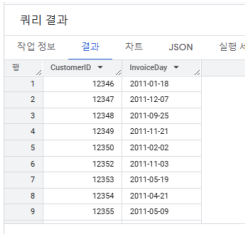
정답코드

```
SELECT
  MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM project_name.modulabs_project.data;
```

왜 over구문인지?

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

```
SELECT
  CustomerID, MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM`white-might-456101-t9.modulabs_project.data`
group by CustomerID
```



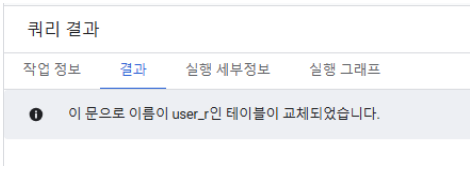
- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

```
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `white-might-456101-t9.modulabs_project.data`
  GROUP BY CustomerID
);
```

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

```
CREATE OR REPLACE TABLE `white-might-456101-t9.modulabs_project.user_r` AS
WITH rd AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `white-might-456101-t9.modulabs_project.data`
  GROUP BY CustomerID
),
rd2 AS (
  SELECT
    MAX(DATE(InvoiceDate)) AS most_recent_date
  FROM `white-might-456101-t9.modulabs_project.data`
)
SELECT
  rd.CustomerID,
  rd.InvoiceDay,
```

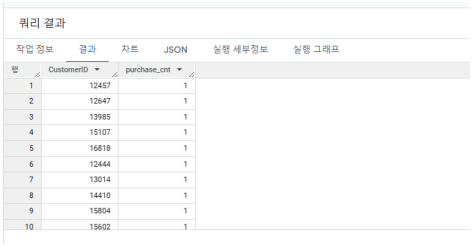
```
 rd2.most_recent_date,
DATE_DIFF(rd2.most_recent_date, rd.InvoiceDay, DAY) AS recency
FROM rd, rd2;
```



Frequency

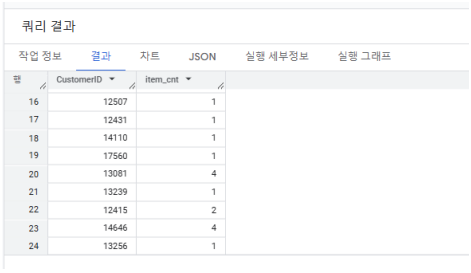
- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
 CustomerID,
 count(distinct invoiceNo) AS purchase_cnt
FROM `white-might-456101-t9.modulabs_project.data`
group by CustomerID;
```



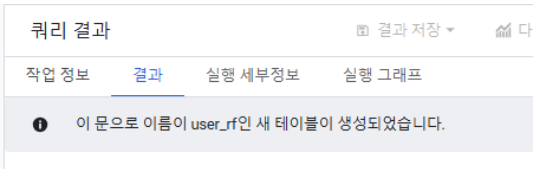
- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
 CustomerID, SUM(Quantity) AS item_cnt
FROM `white-might-456101-t9.modulabs_project.data`
group by CustomerID;
```



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

```
CREATE OR REPLACE TABLE white-might-456101-t9.modulabs_project.user_rf AS
WITH purchase_cnt AS (
SELECT
  CustomerID,
  count(distinct invoiceNo) AS purchase_cnt
FROM `white-might-456101-t9.modulabs_project.data`
group by CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID, SUM(Quantity) AS item_cnt
FROM `white-might-456101-t9.modulabs_project.data`
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 `white-might-456101-t9.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```



Monetary

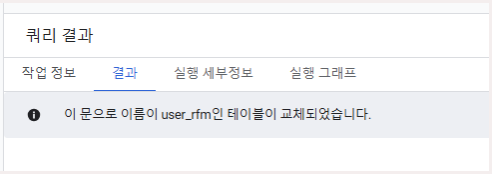
- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT CustomerID,
ROUND(SUM(UnitPrice), 1) AS user_total
FROM `white-might-456101-t9.modulabs_project.data`
group by CustomerID;
```

행	CustomerID	user_total
1	12346	2.1
2	12347	481.2
3	12348	18.7
4	12349	305.1
5	12350	25.3
6	12352	330.5
7	12353	24.3
8	12354	261.2
9	12355	54.7

- **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 white-might-456101-t9.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 white-might-456101-t9.modulabs_project.user_rf rf
LEFT JOIN (
SELECT CustomerID,
ROUND(SUM(UnitPrice), 1) AS user_total
FROM `white-might-456101-t9.modulabs_project.data`
group by CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```



정답 코드

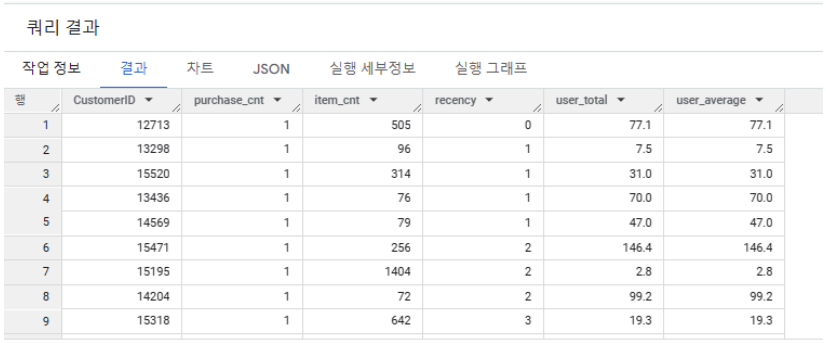
```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
```

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

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select * FROM `white-might-456101-t9.modulabs_project.user_rfm`
```

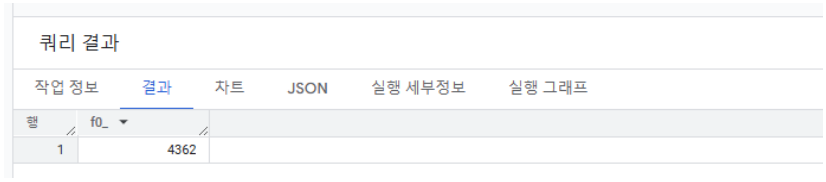


쿼리 결과

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	77.1	77.1
2	13298	1	96	1	7.5	7.5
3	15520	1	314	1	31.0	31.0
4	13436	1	76	1	70.0	70.0
5	14569	1	79	1	47.0	47.0
6	15471	1	256	2	146.4	146.4
7	15195	1	1404	2	2.8	2.8
8	14204	1	72	2	99.2	99.2
9	15318	1	642	3	19.3	19.3

고유한 CustomerID 수



쿼리 결과

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

행	f0_
1	4362

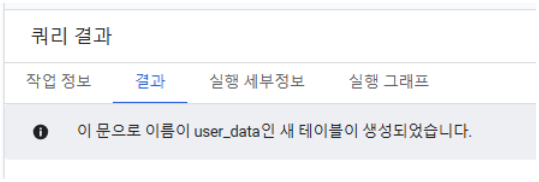
11-8. 추가 Feature 추출

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 `white-might-456101-t9.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `white-might-456101-t9.modulabs_project.data`
  GROUP BY CustomerID
)
```

```
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM white-might-456101-t9.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



정답 코드

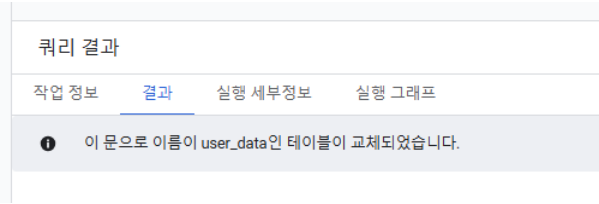
```
CREATE OR REPLACE TABLE `project_name.modulabs_project.user_data` AS
WITH purchase_intervals AS (
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
  FROM (
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, 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;
```

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 `white-might-456101-t9.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
```

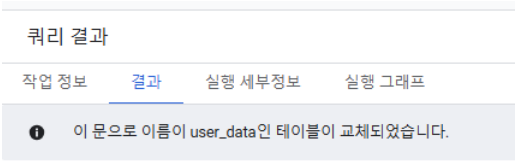
```
    `white-might-456101-t9.modulabs_project.data`
   WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `white-might-456101-t9.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`white-might-456101-t9.modulabs_project.user_data` AS
WITH TransactionInfo AS (
  SELECT
    CustomerID,
count(case when InvoiceNo LIKE 'C%' then 1 end) AS cancel_frequency,
COUNT(InvoiceNo) AS total_transactions
  FROM `white-might-456101-t9.modulabs_project.data`
group by CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), round( (t.cancel_frequency / t.total_transactions)*100, 2) AS cancel_rate
FROM `white-might-456101-t9.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```



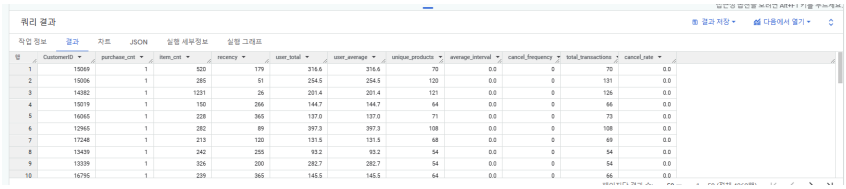
정답코드

```
CREATE OR REPLACE TABLE `project_name.modulabs_project.user_data` AS

WITH TransactionInfo AS (
```

```
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS cancel_frequency
  FROM `project_name.modulabs_project.data`
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(cancel_frequency/total_transactions, 2) AS cancel_rate
FROM `project_name.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 `white-might-456101-t9.modulabs_project.user_data`;
```



회고

Keep :

1. 중간에 잘못된 부분을 인지하고, 정확한 문제 원인 파악 후 해결하려고 한 점.
2. 데이터와 테이블을 모두 삭제하고 새로 생성한 점에서 시간 지체가 있었지만 끝까지 포기하지 않은 점.
3. 다시 푸는 과정에서 RFM 개념과 코드를 더 꼼꼼하게 이해할 수 있었던 점

Problem :

1. UnitPrice가 0이 아닌 데이터로 테이블 생성해야 하는데 0인 데이터로 잘못 넣어서 반이상을 다시 해야했음.
2. 문법적인 개념 학습이 더 필요하고, RFM을 처음 알게 된 개념이라서 이해하고 푸는데 시간이 걸렸다.

Try :

1. 이해가 안 가는 개념은 질문을 통해 해결하고, 정답 코드와 비교하며 코드 리뷰해야겠다.
2. Syntax error부분을 다시 정리해야겠다.
3. 너무 큰 실수는 하지 않도록 해야겠다.