김수민_고객을 세그먼테이션하자 [프로젝트]

김수민_고객을 세그먼테이션하자 [프로젝트]

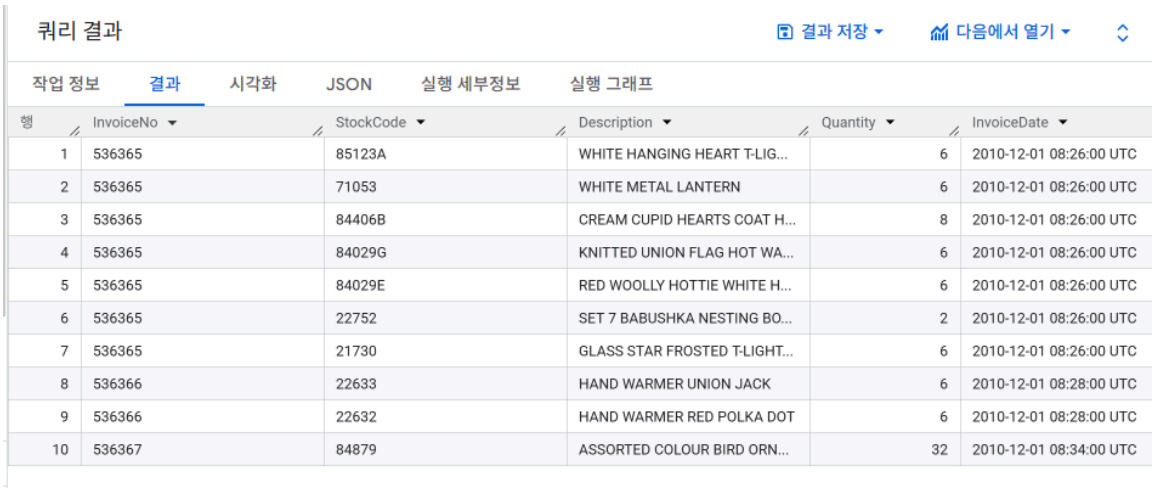
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `natural-system-482801-s7.modulabs_project.data`
LIMIT 10;
```

[결과 이미지를 넣어주세요]



쿼리 결과 — 결과 저장 — 다음에서 열기

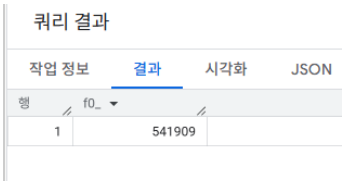
작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보 | 실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM `natural-system-482801-s7.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

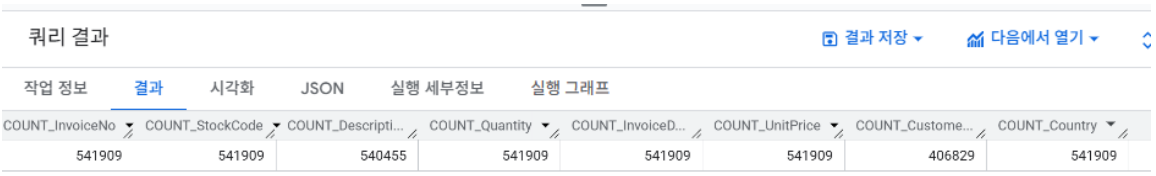


데이터 수 세기

- **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 `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



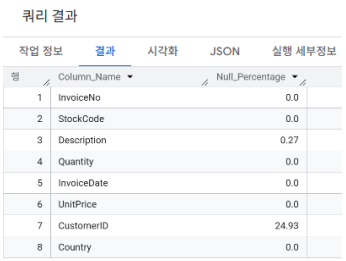
11-4. 데이터 전처리 방법(1): 결측치 제거

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, ROUND((COUNT(*) - COUNT(InvoiceNo)) / COUNT(*) * 100, 2) AS Null_Percentage FROM `natural-system-482801-s7.modulabs_project.data`
UNION ALL
SELECT 'StockCode', ROUND((COUNT(*) - COUNT(StockCode)) / COUNT(*) * 100, 2) FROM `natural-system-482801-s7.modulabs_project.data`
UNION ALL
SELECT 'Description', ROUND((COUNT(*) - COUNT(Description)) / COUNT(*) * 100, 2) FROM `natural-system-482801-s7.modulabs_project.data`
UNION ALL
SELECT 'Quantity', ROUND((COUNT(*) - COUNT(Quantity)) / COUNT(*) * 100, 2) FROM `natural-system-482801-s7.modulabs_project.data`
UNION ALL
SELECT 'InvoiceDate', ROUND((COUNT(*) - COUNT(InvoiceDate)) / COUNT(*) * 100, 2) FROM `natural-system-482801-s7.modulabs_project.data`
UNION ALL
SELECT 'UnitPrice', ROUND((COUNT(*) - COUNT(UnitPrice)) / COUNT(*) * 100, 2) FROM `natural-system-482801-s7.modulabs_project.data`
UNION ALL
SELECT 'CustomerID', ROUND((COUNT(*) - COUNT(CustomerID)) / COUNT(*) * 100, 2) FROM `natural-system-482801-s7.modulabs_project.data`
UNION ALL
SELECT 'Country', ROUND((COUNT(*) - COUNT(Country)) / COUNT(*) * 100, 2) FROM `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



쿼리 결과

작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보

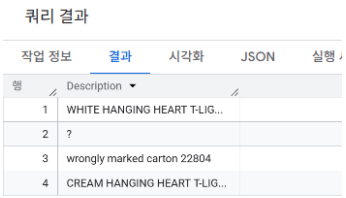
행	Column_Name	Null_Percentage
1	InvoiceNo	0.0
2	StockCode	0.0
3	Description	0.27
4	Quantity	0.0
5	InvoiceDate	0.0
6	UnitPrice	0.0
7	CustomerID	24.93
8	Country	0.0

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM `natural-system-482801-s7.modulabs_project.data`
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]



결측치 처리

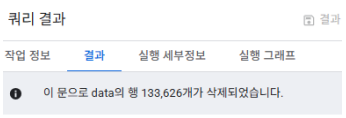
- **DELETE 구문을 사용하며, WHERE 절을 통해 데이터를 제거할 조건을 제시**

```
SELECT COUNT(*)
FROM `natural-system-482801-s7.modulabs_project.data`
WHERE Description IS NULL OR CustomerID IS NULL;
```

1차 삭제 (`Description` 결측치): **1,454개**

2차 삭제 (`CustomerID` 결측치): **133,626개**

합계 : 135,080



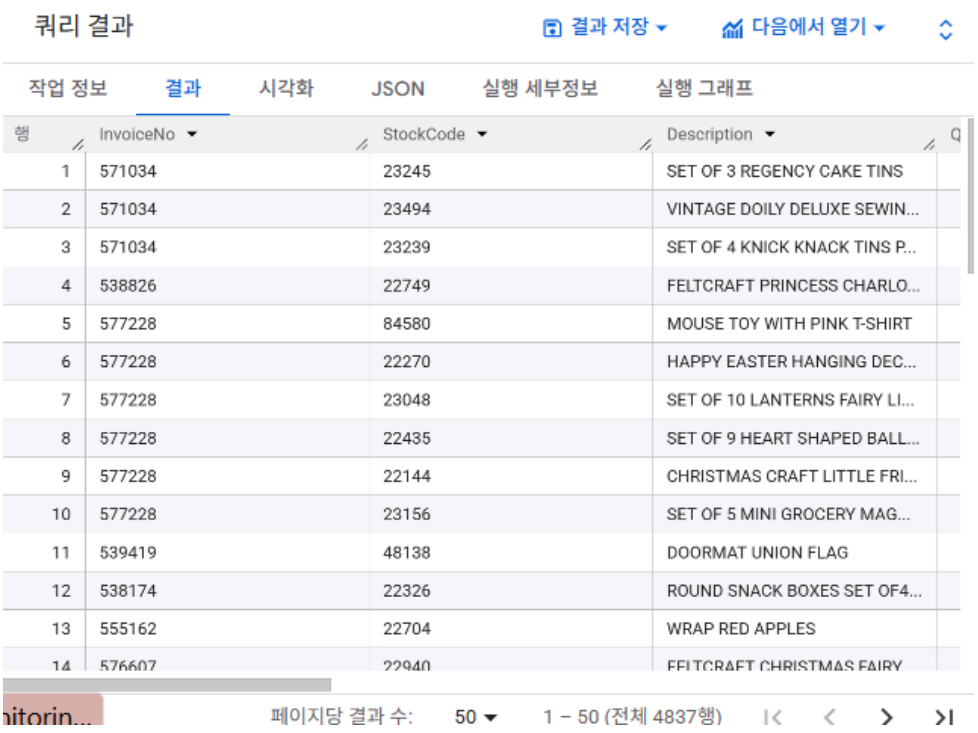
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT
    InvoiceNo, StockCode, Description, Quantity,
    InvoiceDate, UnitPrice, CustomerID, Country,
    COUNT(*) AS cnt
FROM `natural-system-482801-s7.modulabs_project.data`
GROUP BY
    InvoiceNo, StockCode, Description, Quantity,
    InvoiceDate, UnitPrice, CustomerID, Country
HAVING cnt > 1
ORDER BY cnt DESC
```

[결과 이미지를 넣어주세요]

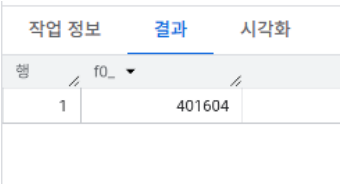


중복값 처리

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

```
CREATE OR REPLACE TABLE `natural-system-482801-s7.modulabs_project.data` AS
SELECT DISTINCT *
FROM `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



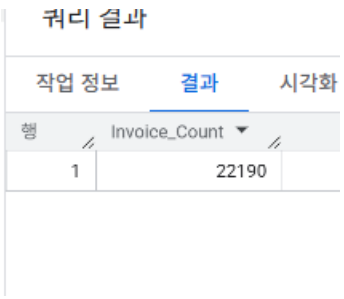
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS Invoice_Count
FROM `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



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

```
SELECT DISTINCT InvoiceNo
FROM `natural-system-482801-s7.modulabs_project.data`
LIMIT 100
```

[결과 이미지를 넣어주세요]

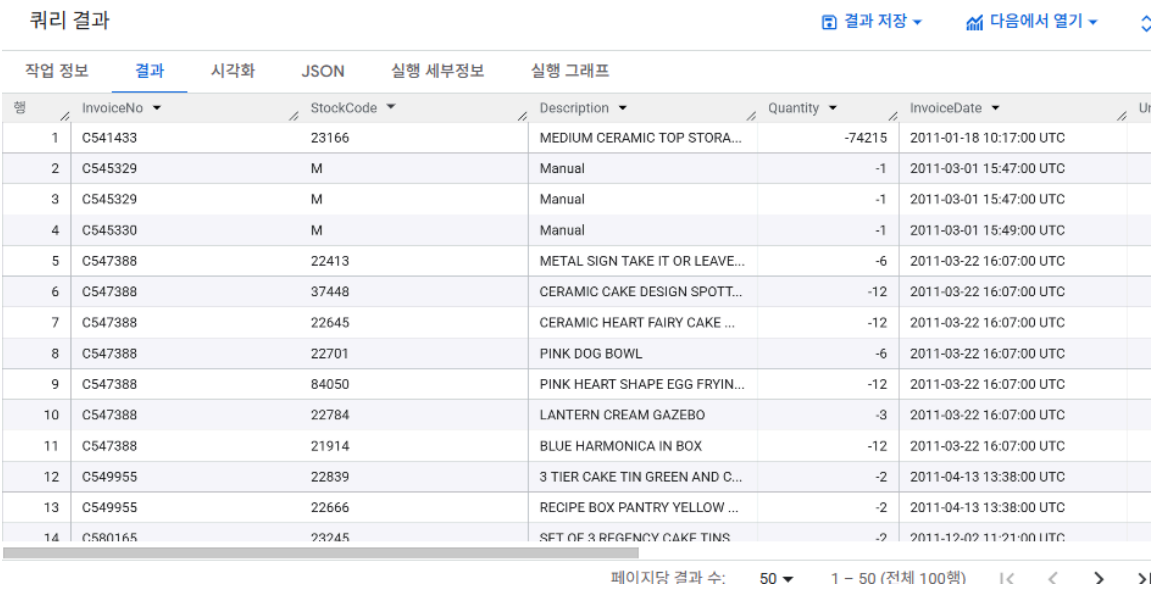
작업 정보 | 결과 | 시각화 | JS

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 `natural-system-482801-s7.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]



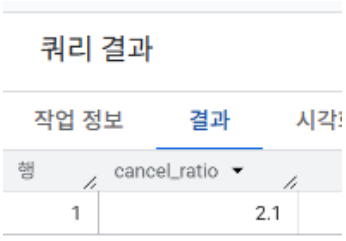
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC
5	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC
6	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC
7	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC
8	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC
9	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC
10	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC
11	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC
12	C549955	22839	3 TIER CAKE TIN GREEN AND C...	-2	2011-04-13 13:38:00 UTC
13	C549955	22666	RECIPE BOX PANTRY YELLOW ...	-2	2011-04-13 13:38:00 UTC
14	C580165	23245	SET OF 3 REGENCY CAKE TINS	-2	2011-12-02 11:21:00 UTC

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT ROUND(
    SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) * 100 /
COUNT(*),
    1
  ) AS cancel_ratio
FROM `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



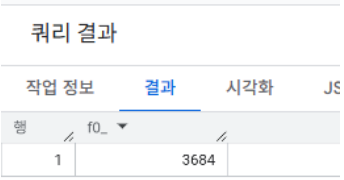
행	cancel_ratio
1	2.1

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT(StockCode))
FROM `natural-system-482801-s7.modulabs_project.data`
```

[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 `natural-system-482801-s7.modulabs_project.data`
WHERE InvoiceNo NOT LIKE 'C%'
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

[결과 이미지를 넣어주세요]

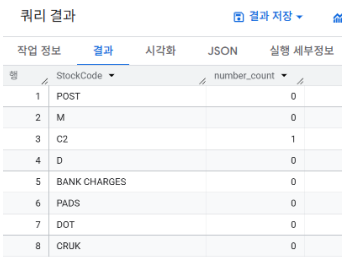
결측치와 중복값을 제거한 상태의 최다판매 결과 (에이펠 예시와 값이 다름)

- **`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 `natural-system-482801-s7.modulabs_project.data`
)
WHERE number_count <= 1;
```

[결과 이미지를 넣어주세요]



쿼리 결과 결과 저장

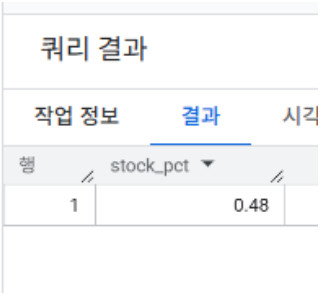
작업 정보 | 결과 | 시각화 | JSON | 실행 세부정보

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)

```
SELECT
   ROUND(
      SUM(CASE
         WHEN (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) <= 1
         THEN 1
         ELSE 0
      END)
      / COUNT(*) * 100
   , 2) AS stock_pct
FROM `natural-system-482801-s7.modulabs_project.data`;
```

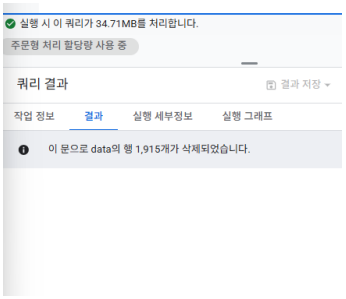
[결과 이미지를 넣어주세요]



- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `natural-system-482801-s7.modulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM `natural-system-482801-s7.modulabs_project.data`
  WHERE (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) <= 1)
```

[결과 이미지를 넣어주세요]



Description 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM `natural-system-482801-s7.modulabs_project.data`
GROUP BY Description
ORDER BY Description_cnt DESC
LIMIT 30
```

[결과 이미지를 넣어주세요]

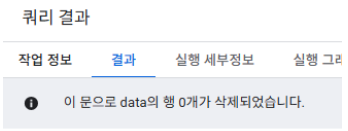
행	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY D...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE ...	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000
14	HEART OF WICKER SMALL	990
15	NATURAL SLATE HEART CHAL...	989
16	JAM MAKING SET WITH JARS	966
17	LUNCH BAG PINK POLKADOT	961
18	LUNCH BAG SUKI DESIGN	932
19	ALARM CLOCK BAKELIKE RED	917
20	WOODEN PICTURE FRAME WHI...	900

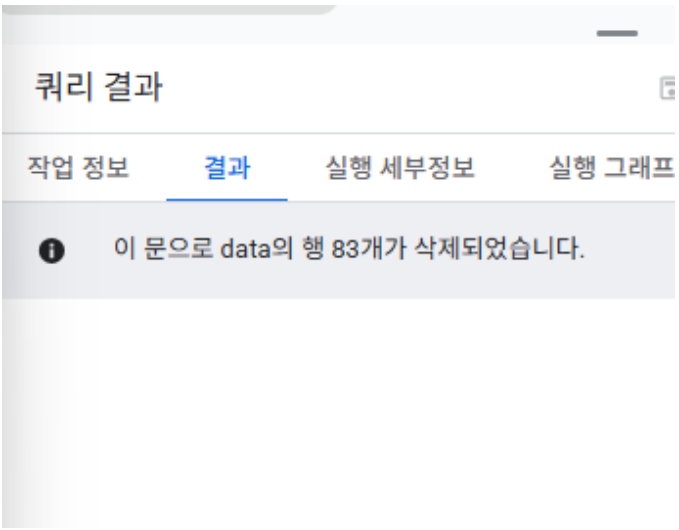
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
-- 1차 시도 (대문자로 통일하지 않아서 변경된 결과 없음)
# DELETE
# FROM `natural-system-482801-s7.modulabs_project.data`
# WHERE Description IN ('NEXT DAY CARRIAGE', 'HIGH RESOLUTION I
MAGE', 'BANK CHARGES', 'POSTAGE');

DELETE
FROM `natural-system-482801-s7.modulabs_project.data`
WHERE UPPER(TRIM(Description)) IN (
```

```
  'NEXT DAY CARRIAGE', 'HIGH RESOLUTION IMAGE', 'BANK CHARGE
S', 'POSTAGE'
);

# 그 외 방법
# 실무 쿼리: 숫자가 5자리가 아닌 모든 것을 한 번에 찾기.
# SELECT DISTINCT StockCode, Description
# FROM `natural-system-482801-s7.modulabs_project.data`
# WHERE NOT REGEXP_CONTAINS(StockCode, r'^[0-9]{5}$');

# 블랙리스트' 키워드 일괄 제거
# DELETE FROM `natural-system-482801-s7.modulabs_project.data`
# WHERE UPPER(Description) REGEXP_CONTAINS(Description, 'POSTA
GE|FEE|CARRIAGE|CHARGE|ADJUST|SAMPLE');
```

[결과 이미지를 넣어주세요]



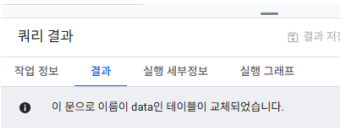


- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `natural-system-482801-s7.modulabs_pro
ject.data` AS
SELECT * EXCEPT (Description),
```

```
        UPPER(Description) AS Description
FROM `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

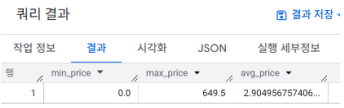


UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
MIN(UnitPrice) AS min_price,
MAX(UnitPrice) AS max_price,
AVG(UnitPrice) AS avg_price
FROM `natural-system-482801-s7.modulabs_project.data`
```

[결과 이미지를 넣어주세요]



- **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
```

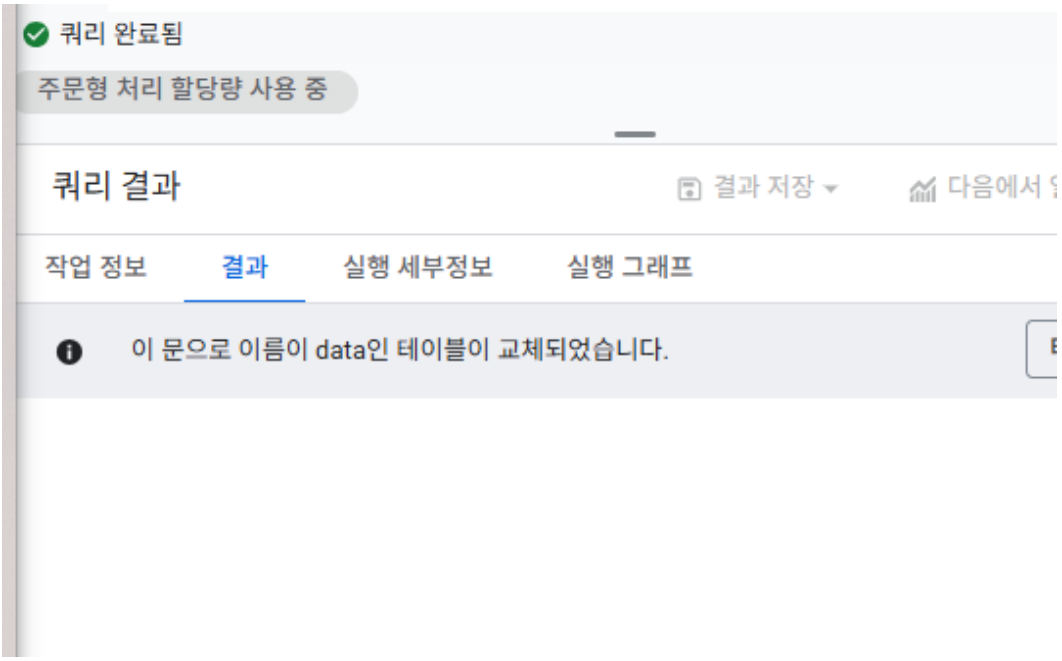
```
FROM `natural-system-482801-s7.modulabs_project.data`
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE `natural-system-482801-s7.modulabs_project.data` AS
SELECT *
FROM `natural-system-482801-s7.modulabs_project.data`
WHERE UnitPrice > 0;
```

[결과 이미지를 넣어주세요]



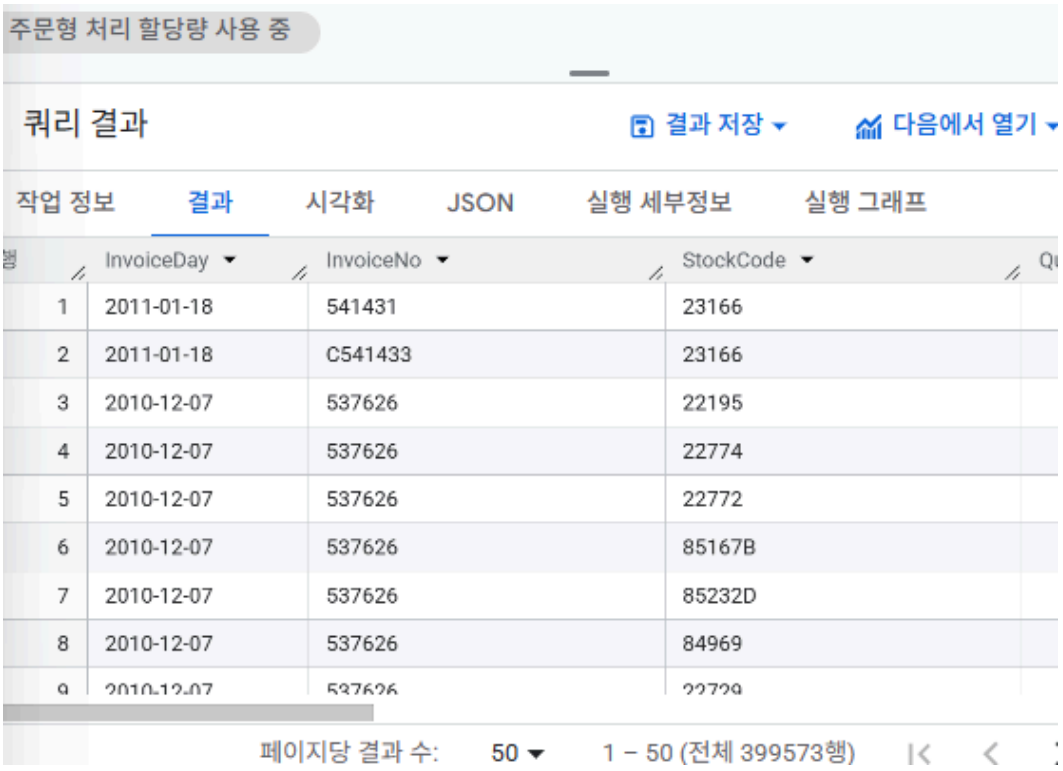
11-7. RFM 스코어

Recency

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

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

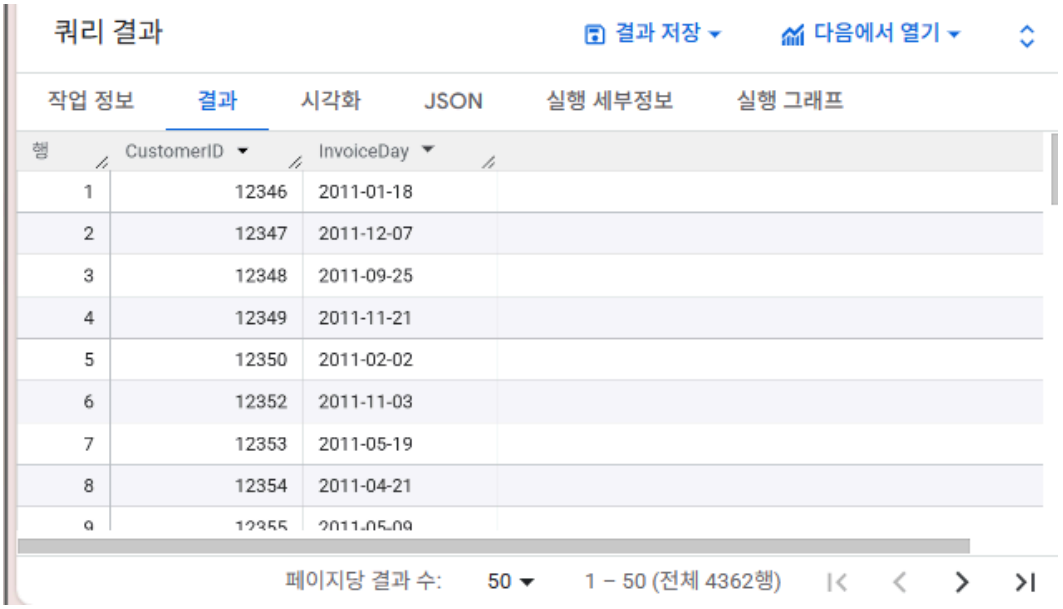
```
  *
FROM `natural-system-482801-s7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

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

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay,
FROM `natural-system-482801-s7.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]



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

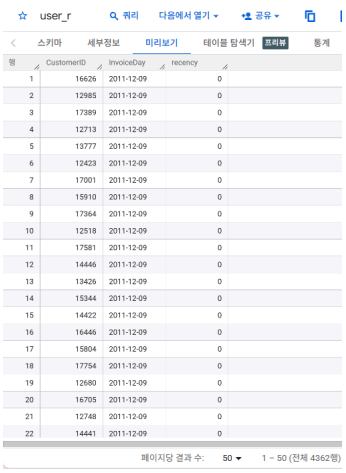
```
SELECT
  CustomerID,
  DATE_DIFF(most_recent_date, InvoiceDay, DAY) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `natural-system-482801-s7.modulabs_project.data`
  GROUP BY CustomerID, InvoiceDate
);
```

[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE `natural-system-482801-s7.modulabs_project.user_r` AS
SELECT
  CustomerID,
  InvoiceDay,
  DATE_DIFF(MAX(InvoiceDay) OVER (), InvoiceDay, DAY) AS recency
FROM (
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `natural-system-482801-s7.modulabs_project.data`
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



행	CustomerID	InvoiceDay	recency
1	16626	2011-12-09	0
2	12985	2011-12-09	0
3	17389	2011-12-09	0
4	12713	2011-12-09	0
5	13777	2011-12-09	0
6	12423	2011-12-09	0
7	17001	2011-12-09	0
8	15910	2011-12-09	0
9	17364	2011-12-09	0
10	12518	2011-12-09	0
11	17581	2011-12-09	0
12	14446	2011-12-09	0
13	13426	2011-12-09	0
14	15344	2011-12-09	0
15	14422	2011-12-09	0
16	16446	2011-12-09	0
17	15804	2011-12-09	0
18	17754	2011-12-09	0
19	12680	2011-12-09	0
20	16705	2011-12-09	0
21	12748	2011-12-09	0
22	14441	2011-12-09	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo)AS purchase_cnt
FROM `natural-system-482801-s7.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 ▾ 다음에서 열기 ▾

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

행	CustomerID	frequency
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

페이지당 결과 수: 50 1 – 50 (전체 4362행)

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `natural-system-482801-s7.modulabs_project.data`
GROUP BY CustomerID;
```

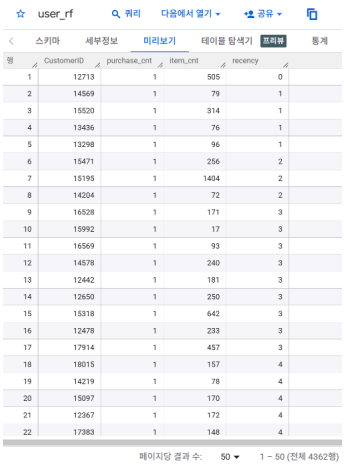
[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf A
S
```

```
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `natural-system-482801-s7.modulabs_project.data`
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
        SELECT
        CustomerID,
            SUM(Quantity) AS item_cnt
        FROM `natural-system-482801-s7.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 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,
        ROUND(SUM(UnitPrice* Quantity), 0) AS user_total
    FROM `natural-system-482801-s7.modulabs_project.data`
GROUP BY CustomerID;
```

[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,
  ROUND(ut.user_total / rf.purchase_cnt, 0) AS user_average
FROM `natural-system-482801-s7.modulabs_project.user_rf` rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
  ROUND(SUM(UnitPrice * Quantity), 0) AS user_total --
  FROM `natural-system-482801-s7.modulabs_project.data`
  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	795.0	795.0
2	13436	1	76	1	197.0	197.0
3	13298	1	96	1	360.0	360.0
4	15520	1	314	1	343.0	343.0
5	14569	1	79	1	227.0	227.0
6	15195	1	1404	2	3861.0	3861.0
7	15471	1	256	2	454.0	454.0
8	14204	1	72	2	151.0	151.0
9	17914	1	457	3	329.0	329.0
10	15318	1	642	3	313.0	313.0
11	16569	1	93	3	124.0	124.0
12	12650	1	250	3	242.0	242.0
13	16528	1	171	3	244.0	244.0
14	15992	1	17	3	42.0	42.0
15	12478	1	233	3	546.0	546.0
16	14578	1	240	3	169.0	169.0
17	12442	1	181	3	144.0	144.0
18	13790	1	748	4	349.0	349.0
19	12367	1	172	4	151.0	151.0
20	16597	1	184	4	90.0	90.0
21	17383	1	148	4	193.0	193.0
22	15097	1	170	4	248.0	248.0

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM `natural-system-482801-s7.modulabs_project.user_rfm`
LIMIT 100
```

[결과 이미지를 넣어주세요]


쿼리 결과

결과 저장 ▾ 다음에서 열기 ▾

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total
1	12713	1	505	0	795
2	13436	1	76	1	197
3	13298	1	96	1	360
4	15520	1	314	1	343
5	14569	1	79	1	227
6	15195	1	1404	2	3861
7	15471	1	256	2	454
8	14204	1	72	2	151
9	17914	1	457	3	329
10	15318	1	642	3	313
11	16569	1	93	3	124
12	12650	1	250	3	242
13	16528	1	171	3	244
14	15992	1	17	3	42
15	12478	1	233	3	546
16	14578	1	240	3	169
17	12442	1	181	3	144
18	13790	1	748	4	349
19	12367	1	172	4	151

페이지당 결과 수: 50 ▾ 1 – 50 (전체 100행)

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 `natural-system-482801-s7.modulabs_project.user_data` AS
WITH unique_products AS (
```

```
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `natural-system-482801-s7.modulabs_project.data`
  GROUP BY CustomerID --
)
SELECT
    ur.*,
    up.* EXCEPT (CustomerID)
FROM `natural-system-482801-s7.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
  ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

user_data 쿼리 다음에서 열기 공유

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프

	CustomerID	purchase_cnt	item_c...	recency	user_total	user_avera...	unique_produ...
1	13302	1	5	155	64.0	64.0	1
2	15940	1	4	311	36.0	36.0	1
3	16738	1	3	297	4.0	4.0	1
4	16078	1	16	283	79.0	79.0	1
5	17291	1	72	308	551.0	551.0	1
6	17948	1	144	147	359.0	359.0	1
7	15118	1	1440	134	245.0	245.0	1
8	17331	1	16	123	175.0	175.0	1
9	13841	1	100	252	85.0	85.0	1
10	15510	1	2	330	250.0	250.0	1
11	18113	1	72	368	76.0	76.0	1
12	16579	1	-12	365	-31.0	-31.0	1
13	13391	1	4	203	60.0	60.0	1
14	13829	1	-12	359	-102.0	-102.0	1
15	15488	1	72	92	76.0	76.0	1
16	15524	1	4	24	440.0	440.0	1
17	16148	1	72	296	76.0	76.0	1
18	16257	1	1	176	22.0	22.0	1
19	16881	1	600	66	432.0	432.0	1
20	17102	1	2	261	26.0	26.0	1
21	17715	1	384	200	326.0	326.0	1
22	17986	1	10	56	21.0	21.0	1

페이지당 결과 수: 50 1 – 50 (전체 4362행)

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 `natural-system-482801-s7.modulabs_projec
t.user_data` AS
WITH purchase_intervals AS (
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(A
VG(interval_), 2) END AS average_interval
  FROM (
    SELECT
      CustomerID,
      DATE_DIFF(DATE(InvoiceDate), LAG(DATE(InvoiceDate)) OVER (PARTITI
ON BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
    FROM
      `natural-system-482801-s7.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `natural-system-482801-s7.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

Customer...	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...
17715	1	384	200	326.0	326.0	1	0.0
13188	1	24	11	100.0	100.0	1	0.0
15940	1	4	311	36.0	36.0	1	0.0
13747	1	8	373	80.0	80.0	1	0.0
15195	1	1404	2	3861.0	3861.0	1	0.0
15313	1	25	110	52.0	52.0	1	0.0
16990	1	100	218	179.0	179.0	1	0.0
16454	1	2	64	6.0	6.0	1	0.0
18068	1	6	289	102.0	102.0	1	0.0
17291	1	72	308	551.0	551.0	1	0.0
15488	1	72	92	76.0	76.0	1	0.0
18174	1	50	7	104.0	104.0	1	0.0
17763	1	12	263	15.0	15.0	1	0.0
13307	1	4	120	15.0	15.0	1	0.0
16144	1	16	246	175.0	175.0	1	0.0
13391	1	4	203	60.0	60.0	1	0.0
16428	1	-1	81	-3.0	-3.0	1	0.0
17307	1	-144	365	-153.0	-153.0	1	0.0
17382	1	24	65	50.0	50.0	1	0.0
15389	1	400	172	500.0	500.0	1	0.0
17956	1	1	249	13.0	13.0	1	0.0
13185	1	12	267	71.0	71.0	1	0.0

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 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 E
ND) AS cancel_frequency
  FROM `natural-system-482801-s7.modulabs_project.data`
  GROUP BY CustomerID
)
```

```
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(SAFE_DIVIDE(t.cancel_frequ
ency, t.total_transactions), 2) AS cancel_rate
FROM `natural-system-482801-s7.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

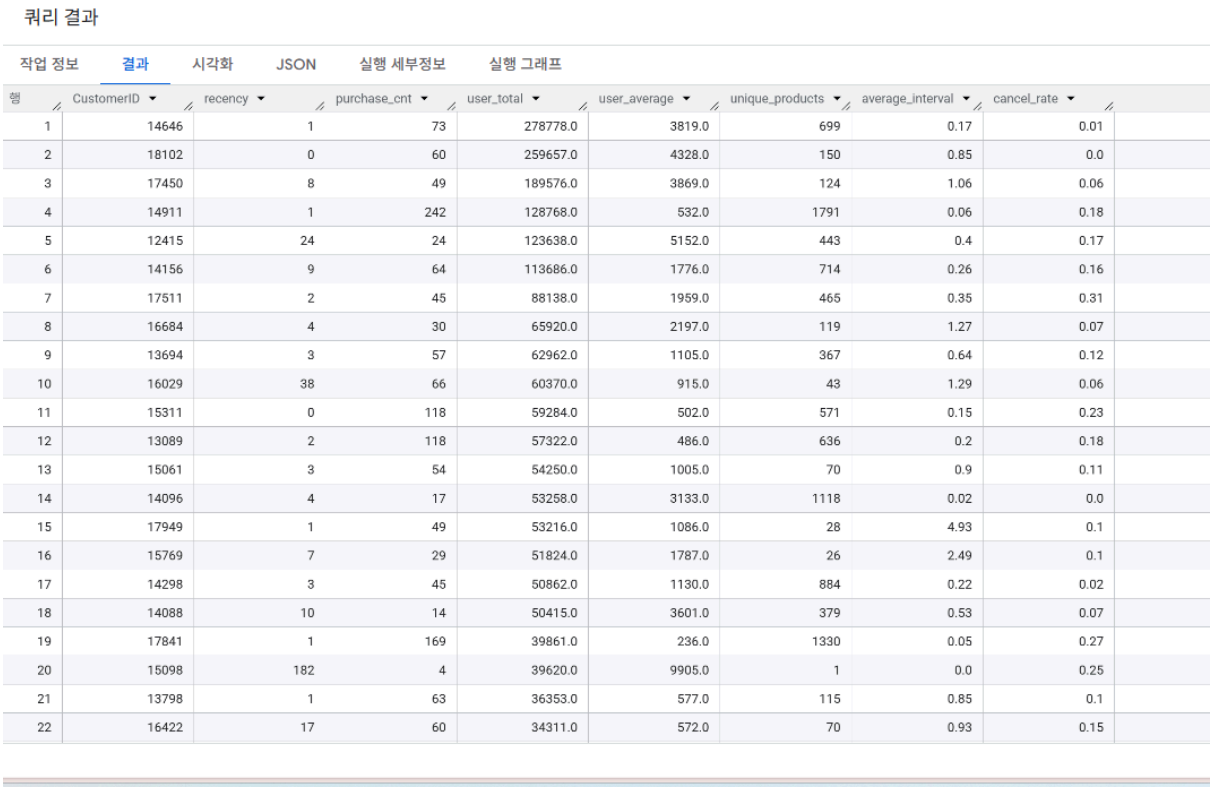
	Customer...	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...	total_transac...	cancel_frequ...	cancel_rate
1	13188	1	24	11	100.0	100.0	1	0.0	1	0	0.0
2	13302	1	5	155	64.0	64.0	1	0.0	1	0	0.0
3	13120	1	12	238	31.0	31.0	1	0.0	1	0	0.0
4	13967	1	42	145	81.0	81.0	2	0.0	1	0	0.0
5	14124	1	1618	84	420.0	420.0	4	0.0	1	0	0.0
6	14582	1	73	40	114.0	114.0	6	0.0	1	0	0.0
7	17524	1	90	361	145.0	145.0	6	0.0	1	0	0.0
8	17022	1	108	31	71.0	71.0	7	0.0	1	0	0.0
9	17647	1	30	65	133.0	133.0	7	0.0	1	0	0.0
10	16178	1	94	138	198.0	198.0	8	0.0	1	0	0.0
11	17117	1	68	288	116.0	116.0	9	0.0	1	0	0.0
12	15523	1	132	84	412.0	412.0	10	0.0	1	0	0.0
13	13182	1	201	45	121.0	121.0	11	0.0	1	0	0.0
14	13494	1	162	311	316.0	316.0	18	0.0	1	0	0.0
15	16366	1	256	18	327.0	327.0	18	0.0	1	0	0.0
16	18203	1	43	157	160.0	160.0	19	0.0	1	0	0.0
17	14894	1	265	85	656.0	656.0	23	0.0	1	0	0.0
18	12833	1	156	145	417.0	417.0	24	0.0	1	0	0.0
19	15585	1	320	176	455.0	455.0	31	0.0	1	0	0.0
20	16666	1	233	19	119.0	119.0	32	0.0	1	0	0.0
21	15790	1	113	10	219.0	219.0	34	0.0	1	0	0.0
22	16823	1	136	227	358.0	358.0	34	0.0	1	0	0.0
23	17051	1	179	194	241.0	241.0	40	0.0	1	0	0.0
24	16971	1	72	38	357.0	357.0	40	0.0	1	0	0.0

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

```
SELECT
  CustomerID,
  recency,
  purchase_cnt,
  user_total,
  user_average,
```

```
  unique_products,
  average_interval,
  cancel_rate
FROM `natural-system-482801-s7.modulabs_project.user_data`
WHERE user_total > 0
ORDER BY user_total DESC, recency ASC;
```

[결과 이미지를 넣어주세요]



쿼리 결과

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

행	CustomerID	recency	purchase_cnt	user_total	user_average	unique_products	average_interval	cancel_rate
1	14646	1	73	278778.0	3819.0	699	0.17	0.01
2	18102	0	60	259657.0	4328.0	150	0.85	0.0
3	17450	8	49	189576.0	3869.0	124	1.06	0.06
4	14911	1	242	128768.0	532.0	1791	0.06	0.18
5	12415	24	24	123638.0	5152.0	443	0.4	0.17
6	14156	9	64	113686.0	1776.0	714	0.26	0.16
7	17511	2	45	88138.0	1959.0	465	0.35	0.31
8	16684	4	30	65920.0	2197.0	119	1.27	0.07
9	13694	3	57	62962.0	1105.0	367	0.64	0.12
10	16029	38	66	60370.0	915.0	43	1.29	0.06
11	15311	0	118	59284.0	502.0	571	0.15	0.23
12	13089	2	118	57322.0	486.0	636	0.2	0.18
13	15061	3	54	54250.0	1005.0	70	0.9	0.11
14	14096	4	17	53258.0	3133.0	1118	0.02	0.0
15	17949	1	49	53216.0	1086.0	28	4.93	0.1
16	15769	7	29	51824.0	1787.0	26	2.49	0.1
17	14298	3	45	50862.0	1130.0	884	0.22	0.02
18	14088	10	14	50415.0	3601.0	379	0.53	0.07
19	17841	1	169	39861.0	236.0	1330	0.05	0.27
20	15098	182	4	39620.0	9905.0	1	0.0	0.25
21	13798	1	63	36353.0	577.0	115	0.85	0.1
22	16422	17	60	34311.0	572.0	70	0.93	0.15

회고

[회고 내용을 작성해주세요]

Keep :

데이터를 다룰때에 함수나 수식도 중요하지만 자주 나는 에러는 철자와 괄호 특수기호 처리였다. 단순히 쿼리를 짜는 것보다 불필요한 실수를 줄이는 것이 중요 포인트라고 느꼈다.

데이터 또한 대소문자 통일이나 결측치를 제거하는 등 하나하나의 작업들이 비로소 온전한 결과를 가져온다는것을 크게 체감하였다.

Problem :

예시 코드를 가져올 때 project_name을 내 실제 프로젝트 ID로 바꾸지 않아 반복적인 에러를 겪었다. 단순한 주소 문제인데도 해결 전까지는 꽤 답답했다.

LAG 함수를 사용해 구매 주기를 계산하는 부분이 가장 어려웠다. 특히 '총 지출액'을 구할 때 단순히 단가만 합산하는 것이 아니라 '단가 × 수량'을 해야 한다는 점을 놓쳐 계산 결과가 꼬이기도 했다.

쿼리를 실행했는데 스키마나 미리보기에서 컬럼이 보이지 않을 때 '작업이 잘못된 건가' 하는 불안함이 컸다. 알고 보니 테이블이 제대로 교체되지 않았거나 스크롤 뒤에 숨어있던 사소한것이지만 데이터 환경에 익숙해지는 과정이 필요함을 느꼈다.

Try :

쿼리를 실행하기 전에 프로젝트 경로와 별칭이 정확한지 한 번 더 체크하는 시스템을 구축해봐야겠다.