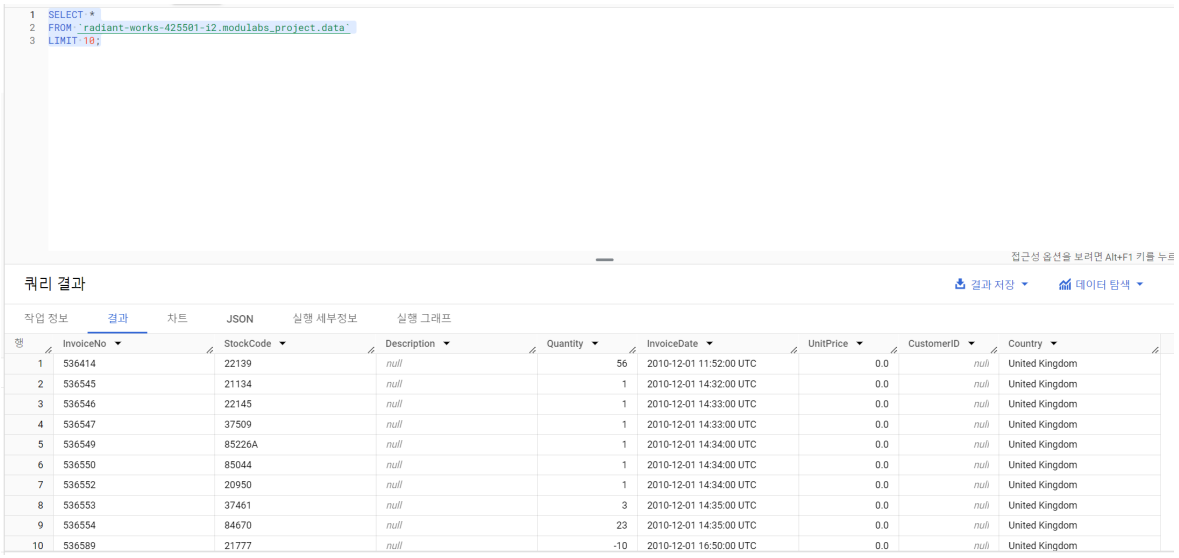
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `radiant-works-425501-i2.modulabs_project.data`
LIMIT 10;
```



```
SELECT *
FROM `radiant-works-425501-i2.modulabs_project.data`
LIMIT 10;
```

접근성 옵션을 보려면 Alt+F1 키를 누르

쿼리 결과

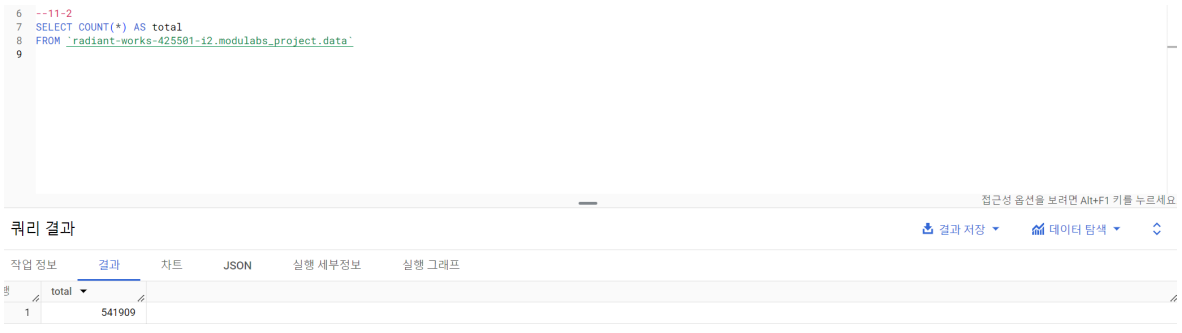
결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	0.0	null	United Kingdom

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*) AS total
FROM `radiant-works-425501-i2.modulabs_project.data`
```



```
--11-2
SELECT COUNT(*) AS total
FROM `radiant-works-425501-i2.modulabs_project.data`
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과

결과 저장 데이터 탐색

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

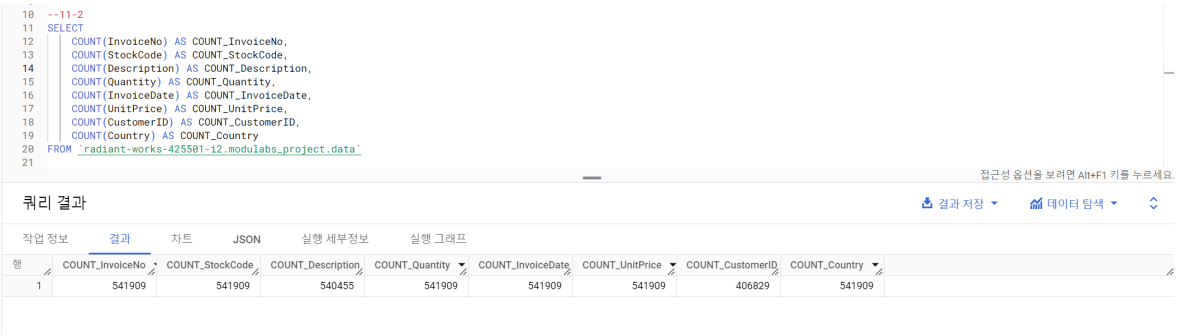
행	total
1	541909

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 `radiant-works-425501-i2.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
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'StockCode' AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'Description' AS column_name,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percenta
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'InvoiceDate' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'UnitPrice' AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'CustomerID' AS column_name,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percen
```

```
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'Country' AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentag
FROM `radiant-works-425501-i2.modulabs_project.data`;
```

```
--11-4
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'StockCode' AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'Description' AS column_name,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `radiant-works-425501-i2.modulabs_project.data`
UNION ALL
SELECT
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `radiant-works-425501-i2.modulabs_project.data`
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과

결과 저장 | 데이터 탐색

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

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

결측치 처리 전략

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

```
SELECT Description
FROM `radiant-works-425501-i2.modulabs_project.data`
WHERE StockCode = '85123A';
```

```
--11-3
SELECT Description
FROM `radiant-works-425501-i2.modulabs_project.data`
WHERE StockCode = '85123A';
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과

결과 저장 | 데이터 탐색

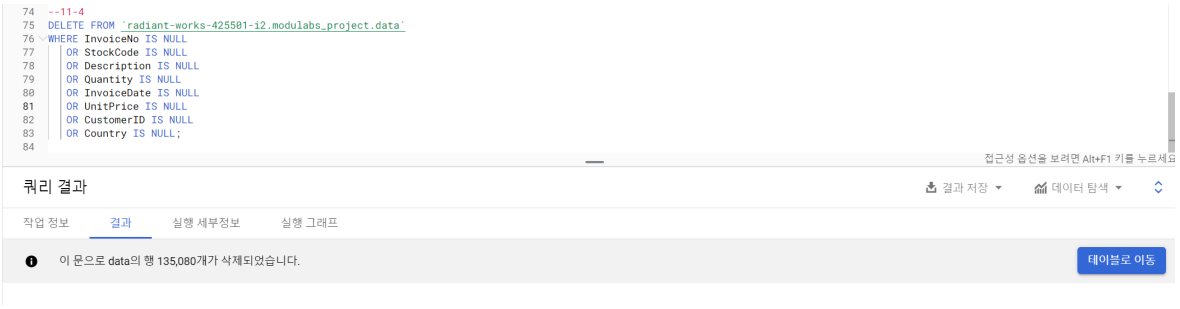
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보 | 실행 그래프

행	Description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	CREAM HANGING HEART T-LIG...
5	CREAM HANGING HEART T-LIG...
6	CREAM HANGING HEART T-LIG...
7	CREAM HANGING HEART T-LIG...
8	CREAM HANGING HEART T-LIG...
9	CREAM HANGING HEART T-LIG...
10	CREAM HANGING HEART T-LIG...
11	CREAM HANGING HEART T-LIG...
12	WHITE HANGING HEART T-LIG...
13	WHITE HANGING HEART T-LIG...
14	WHITE HANGING HEART T-LIG...
15	WHITE HANGING HEART T-LIG...
16	WHITE HANGING HEART T-LIG...
17	WHITE HANGING HEART T-LIG...

결측치 처리

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

```
DELETE FROM `radiant-works-425501-i2.modulabs_project.data`
WHERE InvoiceNo IS NULL
   OR StockCode IS NULL
   OR Description IS NULL
   OR Quantity IS NULL
   OR InvoiceDate IS NULL
   OR UnitPrice IS NULL
   OR CustomerID IS NULL
   OR Country IS NULL;
```



11-5. 데이터 전처리(2): 중복값 처리

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**

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

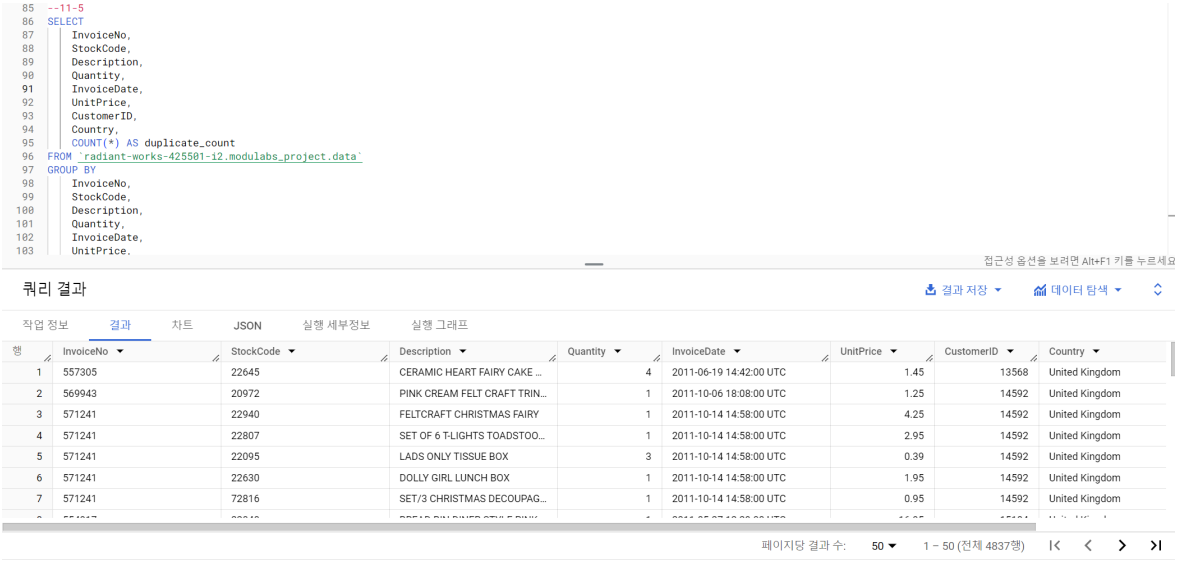
[결과 이미지를 넣어주세요]

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 duplicate_count
FROM `radiant-works-425501-i2.modulabs_project.data`
GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
```

```
    Country
HAVING COUNT(*) > 1;
```



```
--11-5
SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS duplicate_count
FROM `radiant-works-425501-i2.modulabs_project.data`
GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과

결과 저장 | 데이터 탐색

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

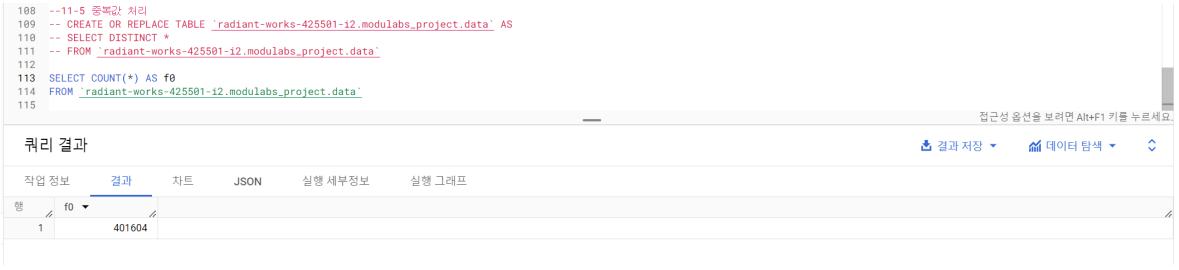
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	557305	22645	CERAMIC HEART FAIRY CAKE …	4	2011-06-19 14:42:00 UTC	1.45	13568	United Kingdom
2	569943	20972	PINK CREAM FELT CRAFT TRIN…	1	2011-10-06 18:08:00 UTC	1.25	14592	United Kingdom
3	571241	22940	FELTCRAFT CHRISTMAS FAIRY	1	2011-10-14 14:58:00 UTC	4.25	14592	United Kingdom
4	571241	22807	SET OF 6 T-LIGHTS TOADSTOO…	1	2011-10-14 14:58:00 UTC	2.95	14592	United Kingdom
5	571241	22095	LADS ONLY TISSUE BOX	3	2011-10-14 14:58:00 UTC	0.39	14592	United Kingdom
6	571241	22630	DOLLY GIRL LUNCH BOX	1	2011-10-14 14:58:00 UTC	1.95	14592	United Kingdom
7	571241	72816	SET/3 CHRISTMAS DECOUPAG…	1	2011-10-14 14:58:00 UTC	0.95	14592	United Kingdom

페이지당 결과 수: 50 | 1 – 50 (전체 4837행)

중복값 처리

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

```
CREATE OR REPLACE TABLE `radiant-works-425501-i2.modulabs_project.data` AS
SELECT DISTINCT *
FROM `radiant-works-425501-i2.modulabs_project.data`
```



```
--11-5 중복값 처리
-- CREATE OR REPLACE TABLE `radiant-works-425501-i2.modulabs_project.data` AS
-- SELECT DISTINCT *
-- FROM `radiant-works-425501-i2.modulabs_project.data`

SELECT COUNT(*) AS f0
FROM `radiant-works-425501-i2.modulabs_project.data`
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과

결과 저장 | 데이터 탐색

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

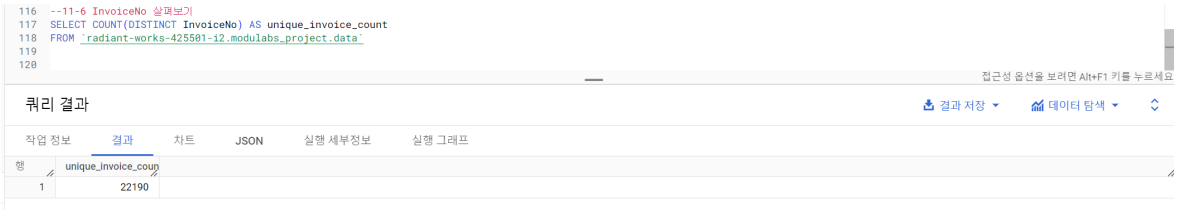
행	f0
1	401604

11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

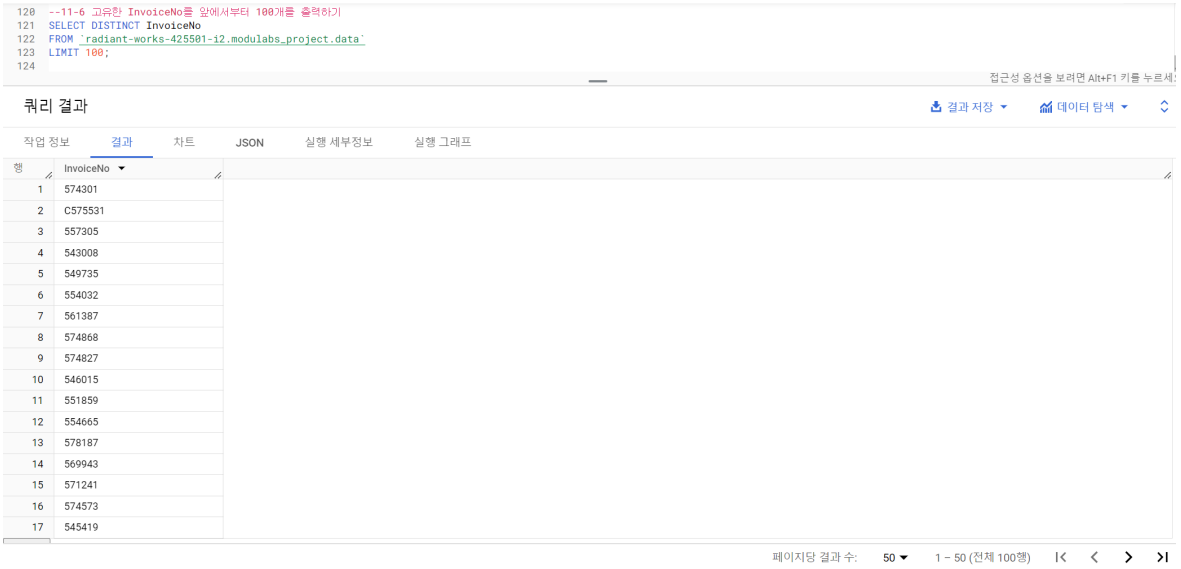
- **고유(unique)한 `InvoiceNo`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `radiant-works-425501-i2.modulabs_project.data`
```



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

```
SELECT DISTINCT InvoiceNo
FROM `radiant-works-425501-i2.modulabs_project.data`
LIMIT 100;
```



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

```
SELECT *
FROM `radiant-works-425501-i2.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

```sql
--11-6 InvoiceNo가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)
SELECT *
FROM `radiant-works-425501-i2.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

접근성 옵션을 보려면 Alt+F1 키를 누르세.

쿼리 결과 | 결과 저장 | 데이터 탐색

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

행	No	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	31	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
2	30	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
3	30	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
4	33	47590A	BLUE HAPPY BIRTHDAY BUNTI…	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	33	47590B	PINK HAPPY BIRTHDAY BUNTI…	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
6	39	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
7	39	84978	HANGING HEART JAR T-LIGHT …	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom
8	39	21485	RETROSPOT HEART HOT WAT…	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom
9	20	21217	RED RETROSPOT ROUND CAK…	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom
10	58	22839	3 TIER CAKE TIN GREEN AND …	-1	2011-03-17 14:24:00 UTC	14.95	14081	United Kingdom
11	58	21534	DAIRY MAID LARGE MILK JUG	-1	2011-03-17 14:24:00 UTC	4.95	14081	United Kingdom
12	53	22699	ROSES REGENCY TEACUP AN…	-1	2011-01-26 17:16:00 UTC	2.95	14849	United Kingdom
13	34	21467	CHERRY CROCHET FOOD COV…	-1	2011-05-17 15:15:00 UTC	3.75	14849	United Kingdom
14	96	22909	SET OF 20 VINTAGE CHRISTM…	-12	2011-10-13 12:02:00 UTC	0.85	14849	United Kingdom
15	96	23376	PACK OF 12 VINTAGE CHRIST…	-24	2011-10-13 12:02:00 UTC	0.39	14849	United Kingdom

페이지당 결과 수: 50 | 1 – 50 (전체 100행)

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

```sql
SELECT ROUND((SUM(CASE WHEN InvoiceNo IS NOT NULL AND LEFT(InvoiceNo, 1) = 'C' THEN 1 ELSE 0 END) /
FROM `radiant-works-425501-i2.modulabs_project.data`
```

```sql
--11-6
SELECT ROUND((SUM(CASE WHEN InvoiceNo IS NOT NULL AND LEFT(InvoiceNo, 1) = 'C' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 1) AS canceled
FROM `radiant-works-425501-i2.modulabs_project.data`
```

접근성 옵션을 보려면 Alt+F1 키를 누르세.

쿼리 결과 | 결과 저장 | 데이터 탐색

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

행	canceled
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```sql
SELECT COUNT(DISTINCT StockCode) AS stockcode_count
FROM `radiant-works-425501-i2.modulabs_project.data`
```

```sql
--11-6 stockcode살펴보기
SELECT COUNT(DISTINCT StockCode) AS stockcode_count
FROM `radiant-works-425501-i2.modulabs_project.data`
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과 | 결과 저장 | 데이터 탐색

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

행	stockcode_count
1	3684

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

```sql
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `radiant-works-425501-i2.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

```sql
--11-6
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `radiant-works-425501-i2.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과

결과 저장 | 데이터 탐색

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

행	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
9	22197	1110
10	23203	1108

- **`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

```sql
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `radiant-works-425501-i2.modulabs_project.data`
)
WHERE number_count BETWEEN 0 AND 1;
```

```sql
--11-6
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `radiant-works-425501-i2.modulabs_project.data`
)
WHERE number_count BETWEEN 0 AND 1;
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요.

쿼리 결과

결과 저장 | 데이터 탐색

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

행	StockCode	number_count
1	POST	0
2	M	0
3	PADS	0
4	D	0
5	BANK CHARGES	0
6	DOT	0
7	CRUK	0
8	C2	1

- **`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)

```sql
WITH FilteredData AS (
    SELECT
        *,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS letter_count
    FROM
        `radiant-works-425501-i2.modulabs_project.data`
)
SELECT ROUND((COUNT(*) * 100.0 / (SELECT COUNT(*) FROM FilteredData)), 2) AS percentage
FROM FilteredData
WHERE letter_count BETWEEN 0 AND 1
```

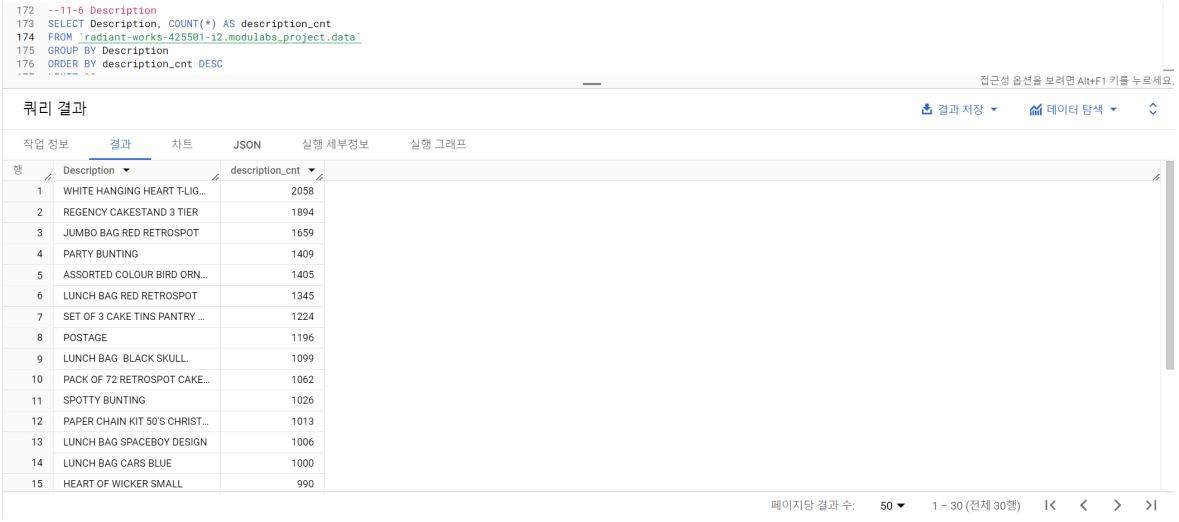
- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `radiant-works-425501-i2.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 `radiant-works-425501-i2.modulabs_project.data`
    )
    WHERE number_count BETWEEN 0 AND 1
)
```

`Description` 살펴보기

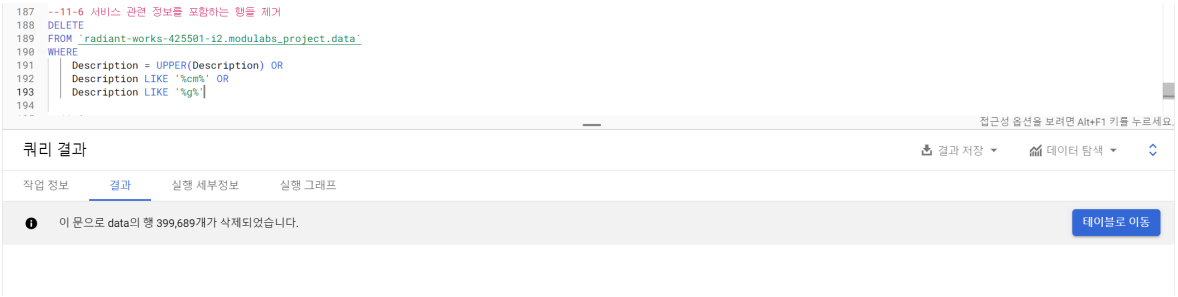
- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS description_cnt
FROM `radiant-works-425501-i2.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```



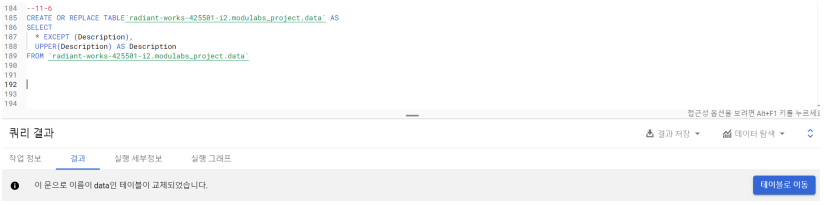
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `radiant-works-425501-i2.modulabs_project.data`
WHERE
    Description = UPPER(Description) OR
    Description LIKE '%cm%' OR
    Description LIKE '%g%'
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE`radiant-works-425501-i2.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `radiant-works-425501-i2.modulabs_project.data`
```



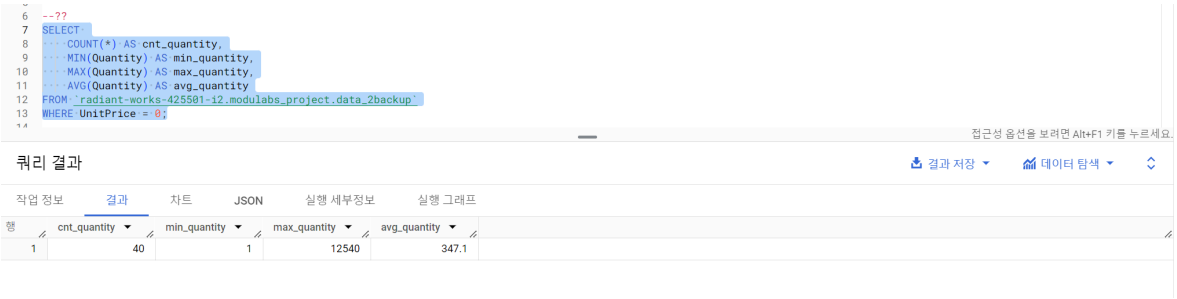
`UnitPrice` 살펴보기

- **`UnitPrice`의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM `radiant-works-425501-i2.modulabs_project.data`
```

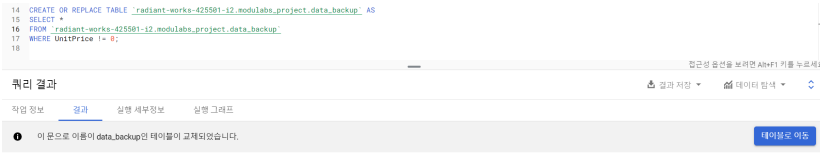
- **단가가 0원인 거래의 개수, 구매 수량(`Quantity`)의 최솟값, 최댓값, 평균 구하기**

```
SELECT
    COUNT(*) AS cnt_quantity,
    MIN(Quantity) AS min_quantity,
    MAX(Quantity) AS max_quantity,
    AVG(Quantity) AS avg_quantity
FROM `radiant-works-425501-i2.modulabs_project.data_2backup`
WHERE UnitPrice = 0;
```



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

```
CREATE OR REPLACE TABLE `radiant-works-425501-i2.modulabs_project.data_backup` AS
SELECT *
FROM `radiant-works-425501-i2.modulabs_project.data_backup`
WHERE UnitPrice != 0;
```

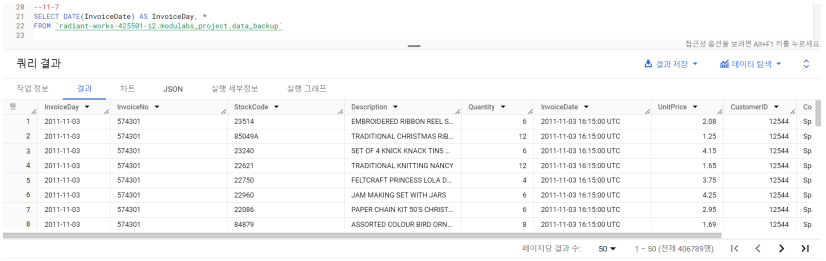


11-7. RFM 스코어

Recency

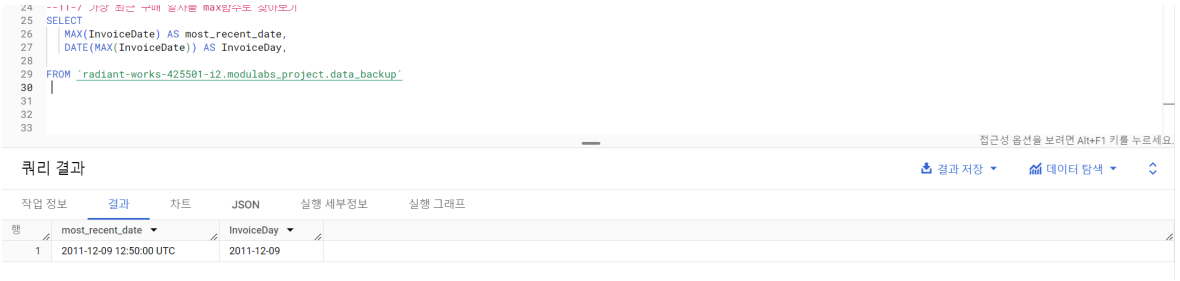
- **`InvoiceDate` 컬럼을 연월일 자료형으로 변경하기**

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `radiant-works-425501-i2.modulabs_project.data_backup`
```



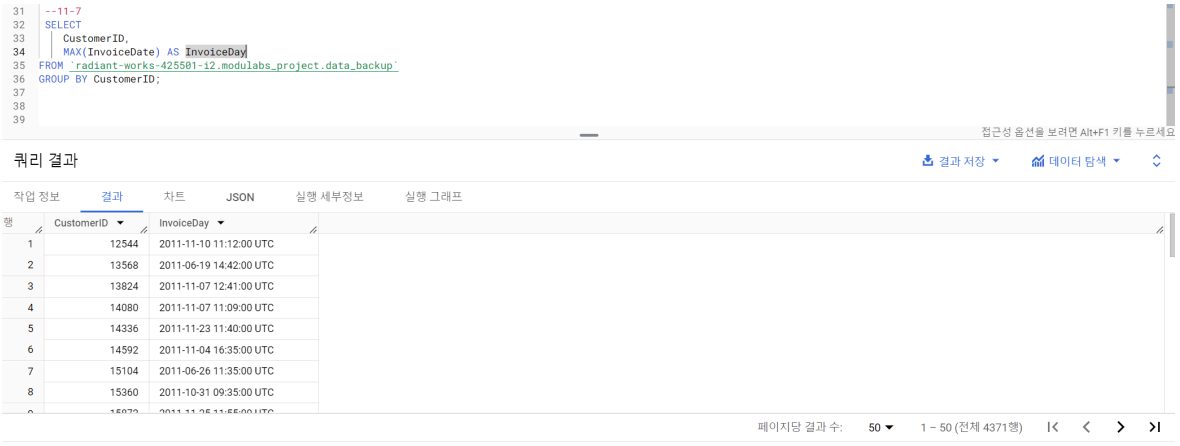
- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

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

FROM `radiant-works-425501-i2.modulabs_project.data_backup`
# *를 추가하면 자꾸 오류가 남,,ㅜㅠ
```



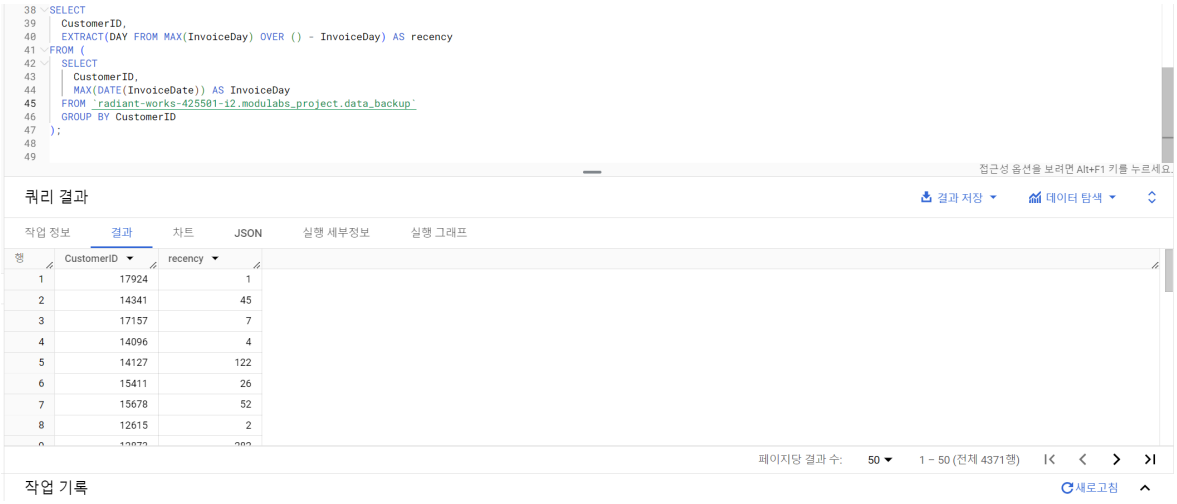
- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

```
 SELECT
    CustomerID,
    MAX(InvoiceDate) AS InvoiceDay
FROM `radiant-works-425501-i2.modulabs_project.data_backup`
GROUP BY CustomerID;
```



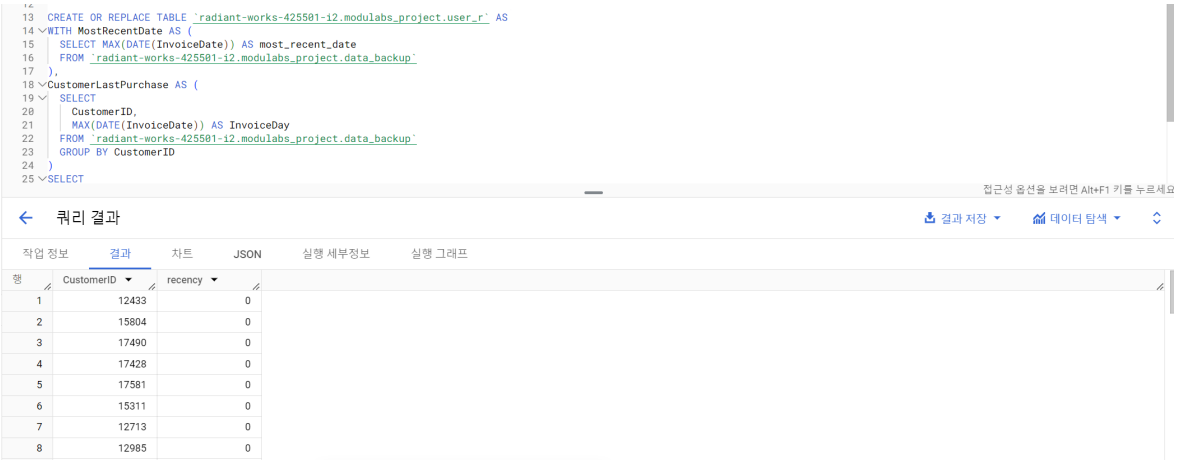
- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

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



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

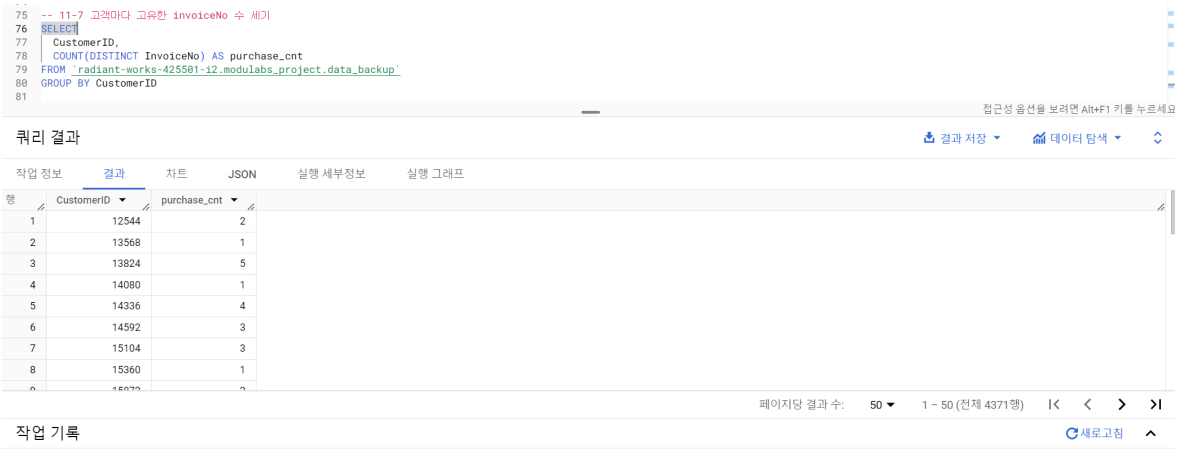
```
CREATE OR REPLACE TABLE `radiant-works-425501-i2.modulabs_project.user_r` AS
WITH MostRecentDate AS (
  SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
  FROM `radiant-works-425501-i2.modulabs_project.data_backup`
),
CustomerLastPurchase AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `radiant-works-425501-i2.modulabs_project.data_backup`
  GROUP BY CustomerID
)
SELECT
  clp.CustomerID,
  DATE_DIFF(mrd.most_recent_date, clp.InvoiceDay, DAY) AS recency
FROM
  CustomerLastPurchase clp
CROSS JOIN
  MostRecentDate mrd;
select *
from `radiant-works-425501-i2.modulabs_project.user_r`
```



Frequency

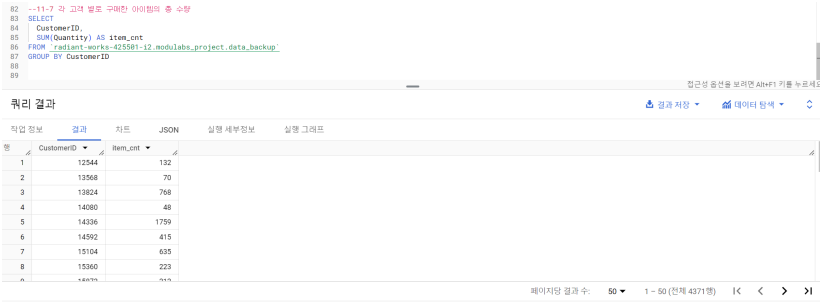
- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `radiant-works-425501-i2.modulabs_project.data_backup`
GROUP BY CustomerID
```



- **각 고객 별로 구매한 아이템의 총 수량 더하기**

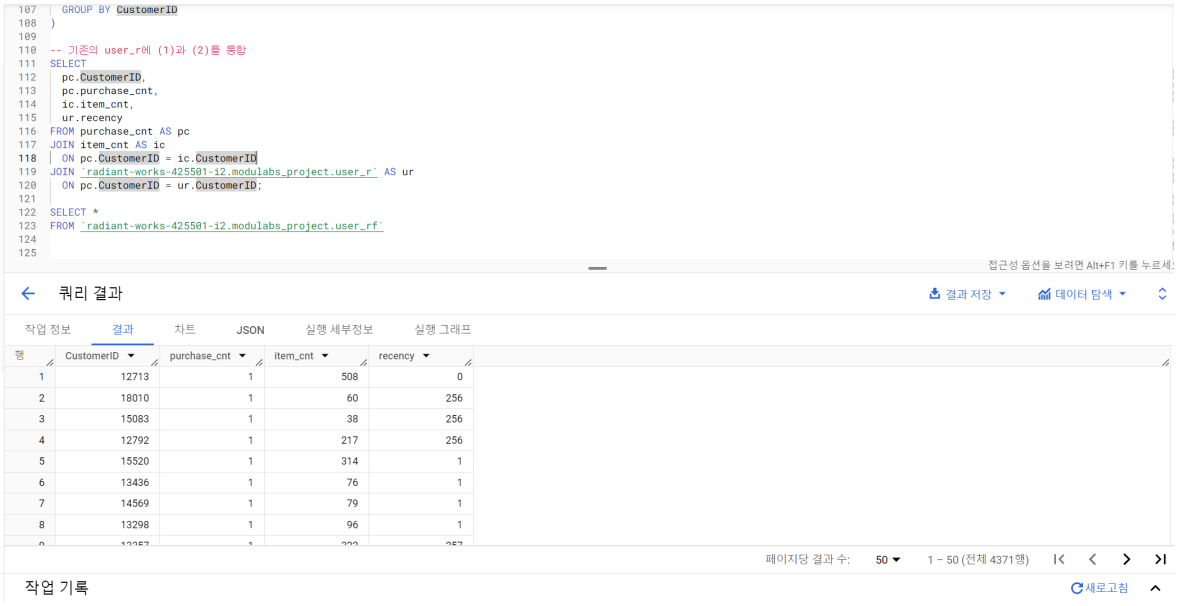
```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `radiant-works-425501-i2.modulabs_project.data_backup`
GROUP BY CustomerID
```



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

```
CREATE OR REPLACE TABLE `radiant-works-425501-i2.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `radiant-works-425501-i2.modulabs_project.data_backup`
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
```

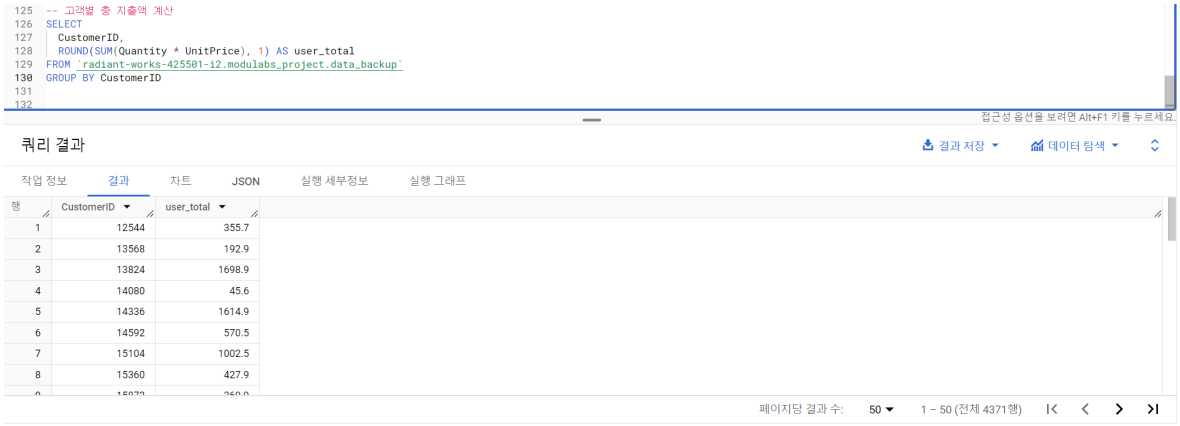
```
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM `radiant-works-425501-i2.modulabs_project.data_backup`
  GROUP BY CustomerID
)

-- 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 `radiant-works-425501-i2.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;

SELECT *
FROM `radiant-works-425501-i2.modulabs_project.user_rf`
```



Monetary

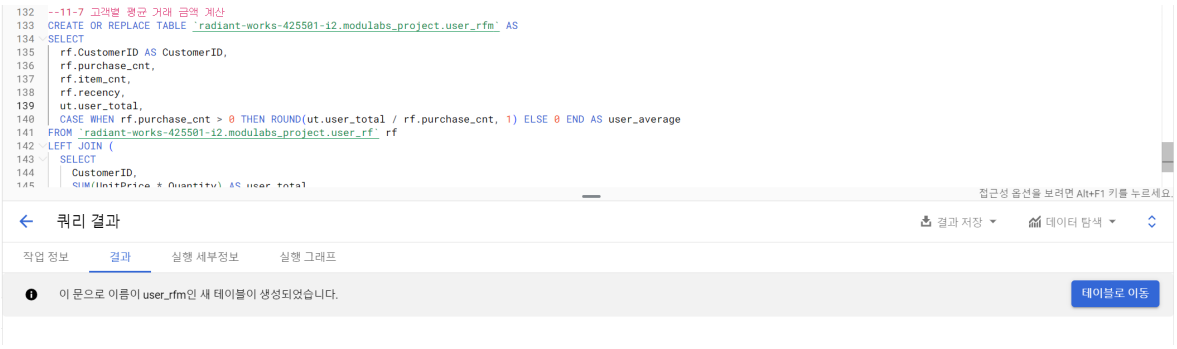
- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `radiant-works-425501-i2.modulabs_project.data_backup`
GROUP BY CustomerID
```



- **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**

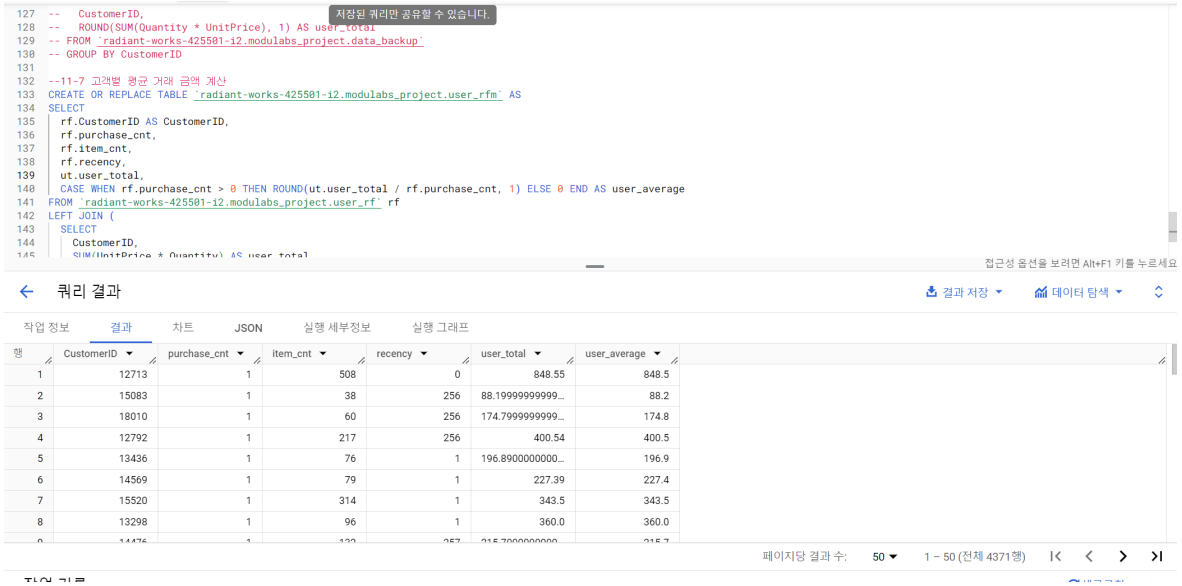
```
CREATE OR REPLACE TABLE `radiant-works-425501-i2.modulabs_project.user_rfm` AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  CASE WHEN rf.purchase_cnt > 0 THEN ROUND(ut.user_total / rf.purchase_cnt, 1) ELSE 0 END AS user_av
FROM `radiant-works-425501-i2.modulabs_project.user_rf` rf
LEFT JOIN (
  SELECT
    CustomerID,
    SUM(UnitPrice * Quantity) AS user_total
  FROM `radiant-works-425501-i2.modulabs_project.data_backup`
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
 FROM radiant-works-425501-i2.modulabs_project.user_rfm
```

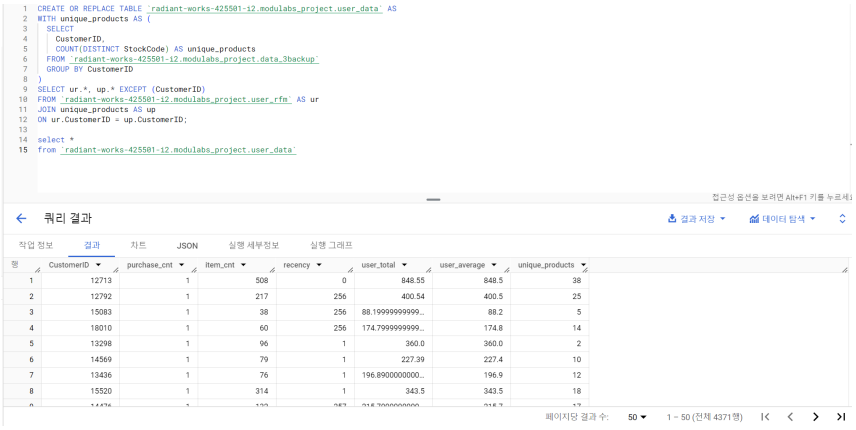


11-8. 추가 Feature 추출

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 `radiant-works-425501-i2.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `radiant-works-425501-i2.modulabs_project.data_3backup`
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `radiant-works-425501-i2.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;

select *
from `radiant-works-425501-i2.modulabs_project.user_data`
```



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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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 radiant-works-425501-i2.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 radiant-works-425501-i2.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.total_transactions, t.cancel_frequency,
       ROUND(t.cancel_frequency / NULLIF(t.total_transactions, 0), 2) AS cancel_rate
FROM `radiant-works-425501-i2.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

```sql
CREATE OR REPLACE TABLE radiant-works-425501-i2.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 radiant-works-425501-i2.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.total_transactions, t.cancel_frequency,
       ROUND(t.cancel_frequency / NULLIF(t.total_transactions, 0), 2) AS cancel_rate
FROM `radiant-works-425501-i2.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
select *
from `radiant-works-425501-i2.modulabs_project.user_data`
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과

결과 저장 | 데이터 탐색

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	purchase_interval	total_transactions	canc
1	13120	1	12	238	30.59999999999…	30.6	1	null	null	null	
2	12505	1	-1	301	-4.5	-4.5	1	null	null	null	
3	16737	1	288	53	417.5999999999…	417.6	1	null	null	null	
4	17986	1	10	56	20.8	20.8	1	null	null	null	
5	14705	1	100	198	179.0	179.0	1	null	null	null	
6	16953	1	10	30	20.8	20.8	1	null	null	null	
7	17102	1	2	261	25.5	25.5	1	null	null	null	
8	13017	1	48	7	204.0	204.0	1	null	null	null	

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

```sql
select *
from `radiant-works-425501-i2.modulabs_project.user_data`
```


```sql
CREATE OR REPLACE TABLE radiant-works-425501-i2.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 radiant-works-425501-i2.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.total_transactions, t.cancel_frequency,
       ROUND(t.cancel_frequency / NULLIF(t.total_transactions, 0), 2) AS cancel_rate
FROM `radiant-works-425501-i2.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
select *
from `radiant-works-425501-i2.modulabs_project.user_data`
```

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과

결과 저장 | 데이터 탐색

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	purchase_interval	total_transactions	canc
1	13120	1	12	238	30.59999999999…	30.6	1	null	null	null	
2	12505	1	-1	301	-4.5	-4.5	1	null	null	null	
3	16737	1	288	53	417.5999999999…	417.6	1	null	null	null	
4	17986	1	10	56	20.8	20.8	1	null	null	null	
5	14705	1	100	198	179.0	179.0	1	null	null	null	
6	16953	1	10	30	20.8	20.8	1	null	null	null	
7	17102	1	2	261	25.5	25.5	1	null	null	null	
8	13017	1	48	7	204.0	204.0	1	null	null	null	

회고...

처음엔 너무 재밌게 하다가 점점 막히고 헤매고 있는 제 자신이 싫어졌던,,ㅎㅎ

제출한 결과가 틀린 게 확실한 건 저도 풀면서 느꼈어요 ㅜㅠㅠ

아직 너무 부족한 실력인 것 같습니당,, 끝나고도 다시 해볼겁니다..!!

이제 더 열심히 해볼게요!!