

고객을 세그먼테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

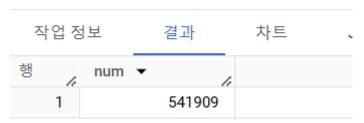
• 테이블에 있는 10개의 행만 출력하기



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

select count(*) as num from `modulabs_project.data` ;

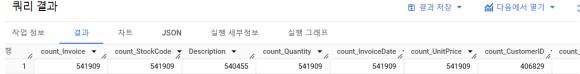
쿼리 결과



데이터 수 세기

• COUNT 함수를 사용해서, 각 컬럼별 데이터 포인트의 수를 세어 보기

```
select
count(InvoiceNo) as count_Invoice,
count(StockCode) as count_StockCode,
count(Description) as Description,
count(Quantity) as count_Quantity,
count(InvoiceDate) as count_InvoiceDate,
count(UnitPrice) as count_UnitPrice,
count(CustomerID) as count_CustomerID,
count(Country) as count_Country
from `modulabs_project.data`;
```



5-4. 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 합치 기

```
SELECT 'InvoiceNo' as column_name,

ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 EN
select 'StokeCode' as column_name, ROUND(SUM(CASE WHEN
FROM modulabs_project.data;
```

-- 첫 행을 invoiceno 라고 이름 지어주고 그 옆의 값을 INVOICENO에 「 --GROUP BY로 진행했으면 GROUP 한 값에 따라 일일이 0,1여부 판단하는

쿼리 결과

작업 정	j보 결과 	차트	JSON	실행 세부정보
행 //	column_name ▼	1.	missing_pe	ercentage
1	InvoiceNo			0.0
2	StokeCode			0.0

결측치 처리 전략

• StockCode = '85123A' 의 Description 을 추출하는 쿼리문을 작성하기

SELECT distinct Description
FROM modulabs_project.data where StockCode = '85123A';

쿼리 결과

작업 정	보 결과 차트
행 //	Description ▼
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG
4	WHITE HANGING HEART T-LIG

결측치 처리

• DELETE 구문을 사용하며, WHERE 절을 통해 데이터를 제거할 조건을 제시

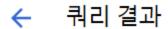
```
UPDATE modulabs_project.data

SET Description = NULL

WHERE Description LIKE '%?%';

delete

FROM modulabs_project.data where Description is NULL or Cu
```



작업 정보 결과 실행 세부정보 실행 그래프

● 이 문으로 data의 행 135,080개가 삭제되었습니다.

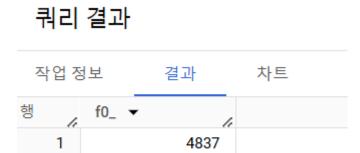
5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT count (*)
FROM(
   select 1 from modulabs_project.data
group by InvoiceNo, StockCode, Description,
```

Quantity, InvoiceDate, UnitPrice, CustomerID, Country having count(*)>1) as ccount;



중복값 처리

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

```
ORDER BY CURRENT_TIMESTAMP()
) AS row_num
FROM modulabs_project.data
)
WHERE row_num = 1;
-- CAST(문자열로 수량) 및 CAST(문자열로 단가):
-- FLOAT64 열(Quantity 및 UnitPrice)을 STRING으로 변환하여 PAR
```

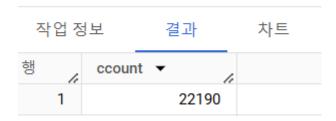
작업 정보 결과 실행 세부정보 실 이 문으로 이름이 data인 테이블이 교체되었습니

5-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

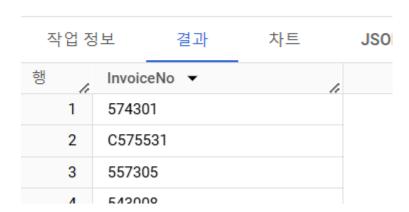
select count(distinct InvoiceNo) as ccount
from modulabs_project.data
where InvoiceNo is not null;



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

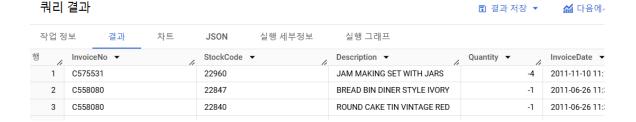
select distinct InvoiceNo as InvoiceNo
from modulabs_project.data
where InvoiceNo is not null limit 100;

쿼리 결과



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

SELECT distinct InvoiceNo as InvoiceNo, StockCode, Descrip FROM modulabs_project.data
WHERE InvoiceNo is not null and InvoiceNo like '%C%'
LIMIT 100;



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

```
with tr as (
SELECT distinct InvoiceNo as InvoiceNo, StockCode, Descr.
FROM modulabs_project.data
WHERE InvoiceNo is not null
)

SELECT
ROUND(
100 * SUM(CASE WHEN InvoiceNo LIKE '%C%' THEN 1 ELS
1
) AS percentage
FROM tr;
-- 이거 with 문 사용하지 않고 더 간단하게 코드 작성 가능할 것 같은데
-- union all 사용해서 진행하는 방법 시도하다가 눈이 아파와서 중도포기
--처음에 sum 내부에 distinct 사용하니 해당 case 값들을 distinct 사
```

쿼리 결과



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

select count(distinct StockCode) as ccount
from modulabs_project.data
where StockCode is not null;

쿼리 결과

작업 정	정보	결과	차트	JSON
행 //	ccount	¥ //		
1		3684		

• 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기

。 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM modulabs_project.data
group by StockCode
ORDER BY sell_cnt DESC
limit 10;
```

쿼리 결과

작업 7	정보	결과	차트	JSON	실행	세부정보	실형
행 //	StockCo	de ▼	//	sell_cnt ▼			
1	85123A				2065		
2	22423				1894		
3	85099B				1659		
4	47566				1409		

• StockCode 의 문자열 내 숫자의 길이를 구해보기

```
WITH UniqueStockCodes AS (
    SELECT DISTINCT StockCode
    FROM modulabs_project.data
)

SELECT
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC;
```

쿼리 결과

작업 정	보 결과	차트	JSON
행 //	number_count ▼	stock_cnt	▼
1	5		3676
2	C)	7
3	1		1

- 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
```

```
FROM modulabs_project.data
)
WHERE number_count between 0 and 1 order by StockCode;
--FROM 값에 nnumber_count를 정의해놔서 where에 쉽게 사용 가능했다.
```

작	업 정	보	결과	차트		JSON	실행서
행	11	StockC	Code ▼		11	number_count	▼ //
	1	BANK	CHARGES				0
	2	C2					1
	3	CRUK					0
	4	D					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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
WITH nc AS (
    SELECT DISTINCT
        StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode)
FROM modulabs_project.data
    WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode))
ORDER BY StockCode
)
SELECT
ROUND(
    100 * COUNT(n.number_count) / COUNT(m.StockCode),
    2
```

```
) AS percentage
FROM modulabs_project.data m
LEFT JOIN nc n ON m.StockCode = n.StockCode;
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM modulabs_project.data
WHERE StockCode IN (
    SELECT DISTINCT StockCode
FROM (
         SELECT DISTINCT
         StockCode,
         LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode)
FROM modulabs_project.data
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode)
ORDER BY StockCode
));
```

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT Description, COUNT(*) AS description_cnt FROM modulabs_project.data group by Description limit 30;

쿼리 결과

작업 정	정보 결과	차트	JSON	실행 세부정보
행 //	Description ▼	1.	description	_cnt ▼
1	ASSORTED COLOU	R BIRD ORN		1405
2	EMBROIDERED RIB	BON REEL S		63
3	EMBROIDERED RIB	BON REEL R		49
4	TDADITIONAL OLID	ICTMAC DID		272

• 대소문자가 혼합된 Description이 있는지 확인하기

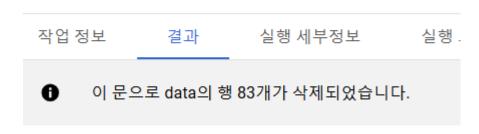
```
SELECT DISTINCT Description
FROM modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
-- 대문자가 있는것은 확인했으니 소문자 있는거것만 분류해서 대소문자 존재
```

작업 정	보 결과	차트	JS
행 //	Description ▼	/	
1	BAG 125g SWIRLY I	MARBLES	
2	3 TRADITIONAL BIS	CUIT CUTTE	
3	BAG 250g SWIRLY I	MARBLES	
4	ESSENTIAL BALM	3.5g TIN IN	

• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE
FROM modulabs_project.data
WHERE
Description like 'Next Day Carriage' or Description like

쿼리 결과



[결과 이미지를 넣어주세요]

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT
 * EXCEPT (Description),
 UPPER(Description) AS Description
FROM
 modulabs_project.data;
-- upper() 함수로 대문자로 바꿔버리기 가능
-- gpt는 그런거 없다고 했던 함수 '* EXCEPT (Description)'... 역사
```

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

● 이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

SELECT min(UnitPrice) AS min_price, max(UnitPrice) AS max_ FROM modulabs_project.data;

즈	1업 정	보	결과	차트	JSON	실행 세부정보	실행 그래프
행	11	min_p	rice ▼	max_price	▼	avg_price ▼	
	1		0.0		649.5	2.904956757405	

• 단가가 0원인 거래의 개수, 구매 수량(quantity)의 최솟값, 최댓값, 평균 구하기

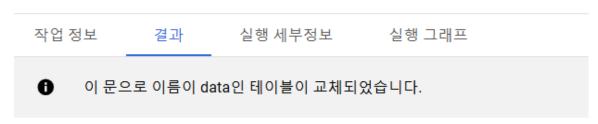
SELECT count(UnitPrice = 0) AS cnt_quantity, min(Quantity)
FROM modulabs_project.data
WHERE UnitPrice = 0.0;

쿼리 결과



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT *
FROM modulabs_project.data
WHERE UnitPrice != 0.0;

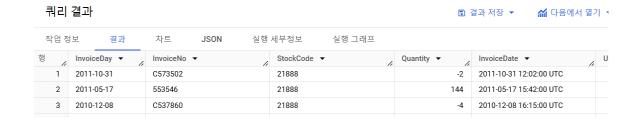


5-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

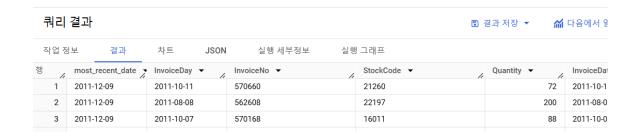
SELECT date(InvoiceDate) AS InvoiceDay, *
FROM modulabs_project.data;



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

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

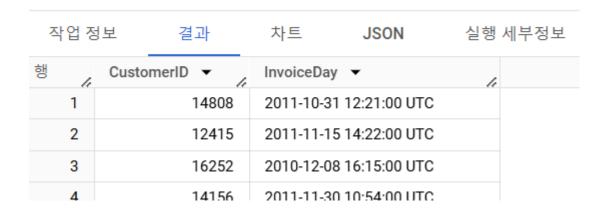
--OVER () 구문을 사용하여 전체 데이터에 대해 가장 최근 날짜를 계산. C -- OVER()는 행을 그룹화하거나 축소하지 않고 집계 함수를 적용하므로 OV -- **GROUP BY**를 사용하면 SQL은 지정된 열을 기준으로 행을 그룹화하.



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT
CustomerID,
max(InvoiceDate) AS InvoiceDay
FROM modulabs_project.data
group by CustomerID;

쿼리 결과



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

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

작업 정	성보 결과	차트	JSON	실행 서
행 //	CustomerID ▼	recency	▼	
1	13805		52	
2	16676		33	
3	12627		10	
4	17799		22	

• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user_r 이라는 이름의 테이블로 저장하기

```
CREATE OR REPLACE TABLE modulabs_project.user_r AS

SELECT

CustomerID,

EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) A:

FROM (

SELECT

CustomerID,

MAX(DATE(InvoiceDate)) AS InvoiceDay

FROM modulabs_project.data
```

```
GROUP BY CustomerID
);
```

⊞ us	er_r q	,쿼리 ▼ +♣공유
스키마	세부정보	미리보기
행 //	CustomerID	recency
1	12526	0
2	18102	0
3	12433	0
4	15311	0
5	17754	0
6	15344	0
7	15910	0
8	12985	0
9	14446	0
10	16705	0

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
count(distinct InvoiceNo) AS purchase_cnt
FROM modulabs_project.data
group by CustomerID ;
```

작업 정	보 결과	차트	JSON
행 //	CustomerID ▼	purchase_cnt	▼ /1
1	14808		14
2	12415		24
3	16252		1
4	14156		64

• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM modulabs_project.data
GROUP BY CustomerID;
```

쿼리 결과

작업 정	g보 결과	차트	JSON
행 //	CustomerID ▼	item_cnt	▼
1	14808		2028
2	12415		76946
3	16252		-158
4	14156		56896

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

```
CREATE OR REPLACE TABLE modulabs_project.user_rf AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
 SELECT
 CustomerID,
  count(distinct InvoiceNo) AS purchase_cnt
FROM modulabs_project.data
group by CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
 SELECT
   CustomerID,
    SUM(Quantity) AS item_cnt
FROM 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 modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```



Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
   CustomerID,
   round(sum(Quantity*UnitPrice),1) AS user_total
FROM modulabs_project.data
group by CustomerID;
```

작업 정	성보 결과	차트	JSON
행 <i>/</i> /	CustomerID ▼	user_total	▼
1	14808		2277.8
2	12415	1	23638.2
3	16252		-295.1
4	14156	1	13685.8

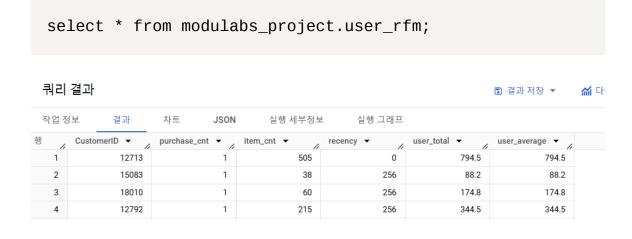
- 고객별 평균 거래 금액 계산
 - 고객별 평균 거래 금액을 구하기 위해 1) data 테이블을 user_rf 테이블과 조인 (LEFT JOIN) 한 후, 2) purchase_cnt 로 나누어서 3) user_rfm 테이블로 저장하기

```
CREATE OR REPLACE TABLE modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
 ut.user_total,
  round(ut.user_total / rf.purchase_cnt, 1) AS user_average
FROM modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
     CustomerID,
  round(sum(Quantity*UnitPrice),1) AS user_total
  FROM modulabs_project.data
 group by CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```



RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기



5-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

1) 고객 별로 구매한 상품들의 고유한 수를 계산하기2)

user_rfm 테이블과 결과를 합치기

3)

user_data 라는 이름의 테이블에 저장하기

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

⊞	use	er_data	Q 쿼리 ▼	★ 보공유 『	복사 🖭 스'	냅샷 📋 삭제	📤 내보내기	▼	
스	키마	세부정보	미리보기	테이블 팀	담색기 <mark>미리보기</mark>	통계	계보 데	이터 프로필	데
행	11	CustomerID	purchase_cnt	item_cnt	recency	user_total //	user_average	unique_products	
	1	14424	1	48	17	322.1	322.1	1	
	2	15657	1	24	22	30.0	30.0	1	
	3	16738	1	3	297	3.8	3.8	1	
	4	17986	1	10	56	20.8	20.8	1	
	5	16138	1	-1	368	-8.0	-8.0	1	
	6	18113	1	72	368	76.3	76.3	1	
	7	17331	1	16	123	175.2	175.2	1	
	R	16323	1	50	196	207 5	207 5	1	

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 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 RO
 FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION
    FROM
     modulabs_project.data
   WHERE CustomerID IS NOT NULL
  )
 GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

⊞ use	er_data	Q 쿼리 ▼	+ 오유	복사 🗷 스님	냅샷 ■ 삭제	📤 내보내기	•		
스키마	세부정보	미리보기	테이블 팀	남색기 <mark>미리보기</mark>	통계	계보 데	이터 프로필	데이터 품질	
행 //	CustomerID	purchase_cnt //	item_cnt	recency	user_total	user_average	unique_products	average_interval	
1	14432	6	2013	9	2248.5	374.8	256	0.2	
2	12428	11	3477	25	6366.0	578.7	256	0.87	
3	13268	14	3525	17	3105.7	221.8	256	0.56	
4	14351	1	12	164	51.0	51.0	1	0.0	
5	13302	1	5	155	63.8	63.8	1	0.0	
6	15488	1	72	92	76.3	76.3	1	0.0	
7	15562	1	39	351	134.6	134.6	1	0.0	
Ω	16222	1	50	106	207.5	207.5	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 modulabs_project.user_data AS

WITH TransactionInfo AS (
   SELECT
        CustomerID,
        count(CustomerID) AS total_transactions,
        sum(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) AS cancel_f
   FROM modulabs_project.data
   where Quantity <0 group by CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), round(t.cancel_frequency
FROM modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON t.CustomerID =u.CustomerID;

-- 이거 맞는거 같은데...
```

작업 정보 결과 실행 세부정보 실행 그래프

• 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 user_data를 출력하기

select * from modulabs_project.user_data;



후기:

전날에 문제 푸는 동안 나타나는 장벽에 열심히 박아가면서 코드를 작성하여서 그런지 그럭 저럭 풀리는 느낌이 듦.

유의미한 데이터를 찾아서 분석하는 기준에 있어서 분석가의 안목이 중요한 부분이라는 것을 알 수 있는 교육과정이였다.