고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

SELECT*

FROM ascendant-nova-456019-r4.modulabs_project2.data LIMIT 10

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT*

FROM ascendant-nova-456019-r4.modulabs_project2.data

[결과 이미지를 넣어주세요]

작업 정	;보 결과 	차트 .	ISON	실행 세부정	보	실행 그래프		
행 //	InvoiceNo ▼	/1	StockCode	•	11	Description •		//
1	536365		85123A			WHITE HANGING	HEART T	LIG
2	536365		71053			WHITE METAL L	ANTERN	
3	536365		84406B			CREAM CUPID H	EARTS CO)AT
4	536365		84029G			KNITTED UNION	FLAG HO	T WA
5	536365		84029E			RED WOOLLY HO	TTIE WHI	TE H
6	536365		22752			SET 7 BABUSHK	A NESTING	G BO
		페	기지당 결과	수: 50 ▼	1 -	50 (전체 54190	9행)	<

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 ascendant-nova-456019-r4.modulabs_project2.data

[결과 이미지를 넣어주세요]

	작업 정	성보 _	결과	차트	JSON	실행 세부정보	실행 그래프		
Č	댈 //	COUNT	Γ_InvoiceNo	COUNT	StockCode	COUNT_Description	COUNT_Quantity	COUNt_InvoiceDate	COUNT
	1		541909		541909	540455	541909	541909	

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

'CustomerID' AS columns_name,

ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COU FROM ascendant-nova-456019-r4.modulabs_project2.data

UNION ALL

SELECT

'Description' AS column_name,

ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COU FROM ascendant-nova-456019-r4.modulabs_project2.data[결과 이미지를 넣어

작업 정보		결과	차트	JSO	N 실행 세-
행 //	colun	nns_name 🔻		mi	ssing_percentage
1	Desc	ription		0.27	
2	Custo	omerID			24.93

결측치 처리 전략

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

SELECT DISTINCT Description FROM ascendant-nova-456019-r4.modulabs_project2.data WHERE StockCode='85123A'

[결과 이미지를 넣어주세요]

작	업정	성보 결과 차트
행	11	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 ascendant-nova-456019-r4.modulabs_project2.data WHERE CustomerID IS NULL OR Description IS NULL

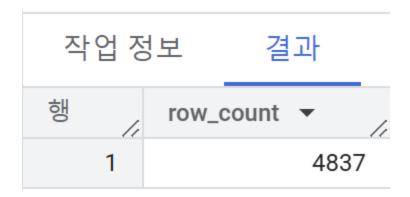


11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
# SELECT 절에서 COUNT(*)의 적용 범위는 서브쿼리에서 반환이된 중복 조합들이 몇:
SELECT COUNT(*) AS row_count
FROM(SELECT
   InvoiceNo,
  StockCode,
  Description,
  Quantity,
  InvoiceDate,
  Unitprice,
  CustomerID,
  Country,
  # 서브쿼리 안의 COUNT()함수의 적용 범위는 GROUP BY 절의 킬럼들 중 중복 조
  Count(*) AS cnt
  FROM ascendant-nova-456019-r4.modulabs_project2.data
  # GROUP BY로 인해 8개의 칼럼이 하나의 조합으로 묶임, 완전히 동일한 행만 같은
  GROUP BY
   InvoiceNo,
    StockCode,
   Description,
   Quantity,
   InvoiceDate,
    Unitprice,
   CustomerID,
   Country
  # HAVING 함수의 조건을 만족하면 중복으로 간주함
   HAVING COUNT(*) > 1)
```



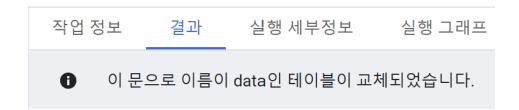
중복값 처리

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

CREATE OR REPLACE TABLE ascendant-nova-456019-r4.modulabs_project2 SELECT DISTINCT *

FROM ascendant-nova-456019-r4.modulabs_project2.data

[결과 이미지를 넣어주세요]

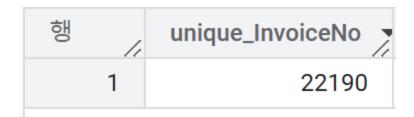


11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT COUNT(DISTINCT InvoiceNo) AS unique_InvoiceNo FROM ascendant-nova-456019-r4.modulabs_project.data



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo FROM ascendant-nova-456019-r4.modulabs_project.data LIMIT 100

행 //	InvoiceNo ▼
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032

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

LIKE 연산자를 사용하여 'C'로 시작하는 값을 필터링함 SELECT*

FROM ascendant-nova-456019-r4.modulabs_project.data WHERE InvoiceNO LIKE 'C%'

.

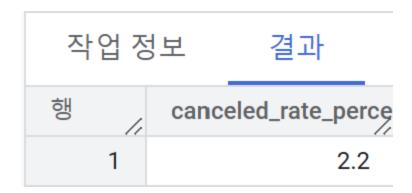


• 구매 건 상태가 Canceled 인 데이터의 비율(%) - 소수점 첫번째 자리까지

SELECT

ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)
/ COUNT(*) * 100, 1) AS canceled_rate_percentage
FROM ascendant-nova-456019-r4.modulabs_project.data

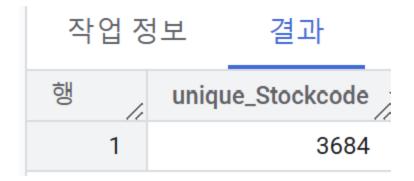
[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode) AS unique_Stockcode FROM ascendant-nova-456019-r4.modulabs_project.data



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

StockCode의 Value 값들을 그룹핑하여 많은 순서대로 출력 SELECT StockCode, COUNT(*) AS sell_cnt FROM ascendant-nova-456019-r4.modulabs_project.data GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10

[결과 이미지를 넣어주세요]

- 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

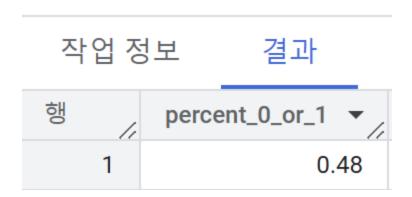
```
SELECT DISTINCT StockCode, number_count
FROM (
    SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')
    FROM ascendant-nova-456019-r4.modulabs_project.data
)
WHERE number_count <= 1
```

[결과 이미지를 넣어주세요]

작업 정	성보 결과	차트 ,	JSON	실행 세
행 //	StockCode ▼	//	number_co	unt 🔻 //
1	POST			0
2	М			0
3	C2			1
4	D			0
5	BANK CHARGES			0

- StockCode 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 숫자가 0~1개인 값들을 가지고있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
SELECT
ROUND(
COUNTIF(LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode,
/ COUNT(*) * 100,
2
) AS percent_0_or_1
FROM ascendant-nova-456019-r4.modulabs_project.data
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM ascendant-nova-456019-r4.modulabs_project.data
WHERE StockCode IN (
SELECT DISTINCT StockCode
FROM (
SELECT StockCode
FROM ascendant-nova-456019-r4.modulabs_project.data
WHERE StockCode IN ('BANK CHARGES', 'POST')
```

) # 삭제 코드를 2번 실행하여 0개가 삭제 되었다는 결과가 출력되었지만 첫번째 코드에

[결과 이미지를 넣어주세요]

작업 정보 결과 실행 세부정보 실행 그래프

① 무으로 data의 행 0개가 삭제되었습니다.

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT

Description,

COUNT(*) AS freq

FROM ascendant-nova-456019-r4.modulabs_project.data

GROUP BY Description

ORDER BY freq DESC

LIMIT 30;

[결과 이미지를 넣어주세요]

작업 정	성보 결과	차트 、	JSON	실행 세
행 //	Description ▼	//	freq ▼	11
1	WHITE HANGING	HEART T-LIG		2058
2	REGENCY CAKEST	TAND 3 TIER		1894
3	JUMBO BAG RED	RETROSPOT		1659
4	PARTY BUNTING			1409
5	ASSORTED COLOU	JR BIRD ORN		1405

• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE FROM ascendant-nova-456019-r4.modulabs_project.data
WHERE Description IN (
   'CARRIAGE',
   'BANK CHARGES',
   'MANUAL',
   'POSTAGE',
   'ADJUSTMENT',
   'SAMPLES',
   'DOTCOM POSTAGE',
   'CHECK'
)
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
6 이문.	으로 data의	행 150개가 삭제되었	었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE ascendant-nova-456019-r4.modulabs_project.c SELECT

* EXCEPT (Description), UPPER(Description) AS Description FROM ascendant-nova-456019-r4.modulabs_project.data

[결과 이미지를 넣어주세요]

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_price,
ROUND(AVG(UnitPrice), 2) AS avg_price
FROM ascendant-nova-456019-r4.modulabs_project.data

[결과 이미지를 넣어주세요]

작업 정	정보	결과	차트	JSON	실행 서	∥부정보
행 //	min_p	orice 🔻	max_	price 🔻	avg_price	~
1		0.	0	38970.0		3.33

• 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, ROUND(AVG(Quantity), 2) AS avg_quantity FROM ascendant-nova-456019-r4.modulabs_project.data WHERE UnitPrice = 0;

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행 cnt_c	quantity 🔻	min_qu	antity 🔻	max_quantity ▼	avg_quantity 🔻
1	40		1	12540	347.1

• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE ascendant-nova-456019-r4.modulabs_project.c
SELECT *
FROM ascendant-nova-456019-r4.modulabs_project.data
WHERE UnitPrice > 0

[결과 이미지를 넣어주세요]

작업 정보 결과 실행 세부정보 실행 그래프

● 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate)

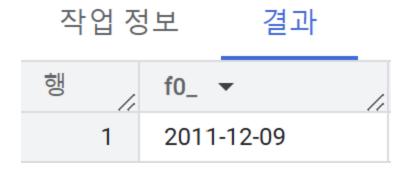
FROM ascendant-nova-456019-r4.modulabs_project.dataSELECT # [[YOUR

[결과 이미지를 넣어주세요]

작	업정	성보 결과 	
행	11	f0_ ▼	/
	1	2011-01-18	
	2	2011-01-18	
	3	2010-12-07	
	4	2010-12-07	
	5	2010-12-07	
	6	2010-12-07	
	7	2010-12-07	

• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT MAX(DATE(InvoiceDate))
FROM ascendant-nova-456019-r4.modulabs_project.data



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT CustomerID, MAX(DATE(InvoiceDate)) AS InvoiceDay FROM ascendant-nova-456019-r4.modulabs_project.data GROUP BY CustomerID;

[결과 이미지를 넣어주세요]

작업 정보		결과		차트	JSON
행 //	Custo	merID 🔻	. //	InvoiceD	ay ▼
1		1234	-6.0	2011-01-	18
2		1234	7.0	2011-12-	07
3		1234	8.0	2011-09-	25
4		1234	9.0	2011-11-	21
5		1235	0.0	2011-02-	02
6		1235	2.0	2011-11-	03
7		1235	3.0	2011-05-	19

• 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 ascendant-nova-456019-r4.modulabs_project.data
GROUP BY CustomerID
)
```

작업 정	, 보 결과	차트	JSON
행 //	CustomerID ▼	recency	~
1	12613.0		31
2	12912.0		2
3	12952.0		5
4	13458.0		7
5	13569.0		17
6	13666.0		60
7	13814.0		56

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

```
CREATE OR REPLACE TABLE `ascendant-nova-456019-r4.modulabs_project.

SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
```

```
FROM `ascendant-nova-456019-r4.modulabs_project.data`
GROUP BY CustomerID
);

SELECT *
FROM `ascendant-nova-456019-r4.modulabs_project.user_r`
LIMIT 100;
```

작업 정	성보 결과	차트	JSON
행 //	CustomerID	recency	· •
1	167	05.0	0
2	158	04.0	0
3	125	26.0	0
4	124	23.0	0
5	174	28.0	0

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS freq FROM ascendant-nova-456019-r4.modulabs_project.data GROUP BY CustomerID

[결과 이미지를 넣어주세요]

작업 정	성보 결	결과	차트		JSON
행 //	Customer	ID ▼	freq	•	//
1		12346.0			2
2		12347.0			7
3		12348.0			4
4		12349.0			1
5		12350.0			1

• 각 고객 별로 구매한 아이템의 총 수량 더하기

SELECT CustomerID, SUM(Quantity)
FROM ascendant-nova-456019-r4.modulabs_project.data
GROUP BY CustomerID

작업 정	성보 결고	<u></u>		JSON
행 //	CustomerID	~	f0_ ▼	//
1	12	346.0		0
2	12	347.0		2458
3	12	348.0		2332
4	12	349.0		630
5	12	350.0		196

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

```
CREATE OR REPLACE TABLE `ascendant-nova-456019-r4.modulabs_project.

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
    SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_cnt
    FROM `ascendant-nova-456019-r4.modulabs_project.data`
    GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
    SELECT CustomerID, SUM(Quantity) AS item_cnt
```

```
FROM `ascendant-nova-456019-r4.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 `ascendant-nova-456019-r4.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 ascendant-nova-456019-r4.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

작업 정	정보 결과	차트	JSON
행 //	CustomerID ▼	user_total	~
1	12346.	.0	0.0
2	12347.	.0	4310.0
3	12348.	.0	1437.2
4	12349.	0	1457.5
5	12350.	0	294.4

• 고객별 평균 거래 금액 계산

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

```
CREATE OR REPLACE TABLE `ascendant-nova-456019-r4.modulabs_project.

WITH monetary_table AS (
    SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS monetary
FROM `ascendant-nova-456019-r4.modulabs_project.data`
GROUP BY CustomerID
)

SELECT
    rf.CustomerID,
    rf.recency,
    rf.purchase_cnt AS frequency,
    rf.item_cnt,
    mt.monetary,
    ROUND(mt.monetary / rf.purchase_cnt, 1) AS avg_order_value
```

FROM `ascendant-nova-456019-r4.modulabs_project.user_rf` AS rf LEFT JOIN monetary_table AS mt ON rf.CustomerID = mt.CustomerID;

[결과 이미지를 넣어주세요]



RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

SELECT *
FROM `ascendant-nova-456019-r4.modulabs_project.user_rfm`
LIMIT 100;

[결과 이미지를 넣어주세요]

작업 정	성보 결과	차트	JSON	실행 세부정보	실행 그래프
행 //	CustomerID ▼	recency	y -	frequency •	item_cnt ▼
1	12713	3.0	0	정렬 메뉴 열기	505
2	13436	5.0	1	1	76
3	14569	9.0	1	1	79
4	15520	0.0	1	1	314
5	13298	3.0	1	1	96
6	15471	0.1	2	1	256
7	15195	5.0	2	1	1404
8	14204	1.0	2	1	72
9	16569	9.0	3	1	93

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 `ascendant-nova-456019-r4.modulabs_project.use

WITH unique_products AS (
    SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
    FROM `ascendant-nova-456019-r4.modulabs_project.data`
    GROUP BY CustomerID
)

SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `ascendant-nova-456019-r4.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

```
작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.
```

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 'ascendant-nova-456019-r4.modulabs_project.use
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(
    DATE(InvoiceDate),
    LAG(DATE(InvoiceDate)) OVER (
     PARTITION BY CAST(CustomerID AS STRING)
     ORDER BY DATE(InvoiceDate)
    ),
    DAY
  ) AS interval_
  FROM 'ascendant-nova-456019-r4.modulabs_project.data'
  WHERE CustomerID IS NOT NULL
 GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `ascendant-nova-456019-r4.modulabs_project.user_data` AS u
```

```
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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 `ascendant-nova-456019-r4.modulabs_project.use

WITH TransactionInfo AS (

SELECT

CustomerID,

COUNT(DISTINCT InvoiceNo) AS total_transactions,

COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS

FROM `ascendant-nova-456019-r4.modulabs_project.data`

WHERE CustomerID IS NOT NULL

GROUP BY CustomerID

)

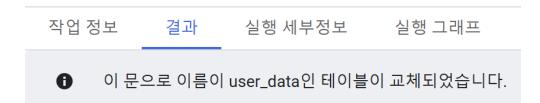
SELECT

u.*,

t.* EXCEPT(CustomerID),
```

ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate FROM `ascendant-nova-456019-r4.modulabs_project.user_data` AS u LEFT JOIN TransactionInfo AS t ON u.CustomerID = t.CustomerID

[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM `ascendant-nova-456019-r4.modulabs_project.user_data`
LIMIT 100;

CustomerID

recency - 마지막 구매로부터 얼마나 시간이 흘렀는지

frequency - 전체 거래 건수

item_cnt - 구매한 아이템 총 수량

monetary - 총 지출 금액

avg_order_value - 평균 주문 금액

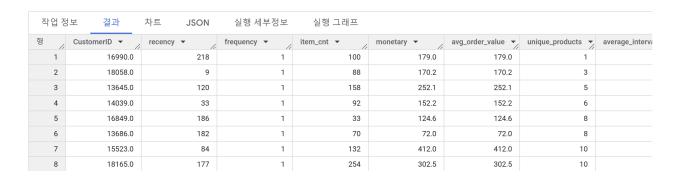
unique_products - 구매한 고유 제품 수

average_interval - 구매 간 평균 일수
```

cancel_frequency - 취소한 거래 횟수

cancel_rate - 전체 거래 중 취소 비율

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

Keep: 정말 어려운 코드가 아니라면 스스로 직접 많이 작성해보려고 했음

Problem: 프로젝트 베이스 러닝이 실력 향상에 많이 도움이 된다는 것은 알고 있으나 오늘과 같은 프로젝트는 양이나 난이도 면에서 많이 내 수준에서는 많이 무리라는 생각이 들었음 또한 어려운 코드는 본인이 직접 작성한 것이 거의 없음

Try : 오늘과 같이 난이도나 시간면에서 부담을 덜 느끼게 되는 프로젝트 베이스 학습을 하게 된다면 더 천천히 여유를 가지고 코드를 뜯어 보면서 학습을 할 것