고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

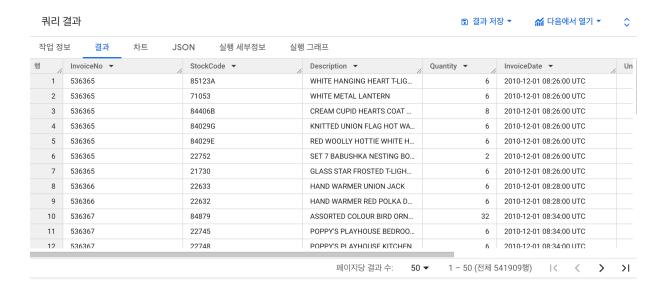
• 테이블에 있는 10개의 행만 출력하기

SELECT *
FROM `vital-program-456100-p3.modulabs_project.data`
LIMIT 10



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT * FROM `vital-program-456100-p3.modulabs_project.data`



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 'vital-program-456100-p3.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_percentage FROM 'vital-program-456100-p3.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 'vital-program-456100-p3.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 'vital-program-456100-p3.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 'vital-program-456100-p3.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_percentage FROM 'vital-program-456100-p3.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_percentage FROM `vital-program-456100-p3.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_percentage FROM `vital-program-456100-p3.modulabs_project.data`

UNION ALL

SELECT

'Country' AS column_name,

 $ROUND (SUM (CASE \ WHEN \ Country \ IS \ NULL \ THEN \ 1 \ ELSE \ 0 \ END) \ / \ COUNT(*) * 100, \ 2) \ AS \ missing_percentage \\ FROM `vital-program-456100-p3.modulabs_project.data`$

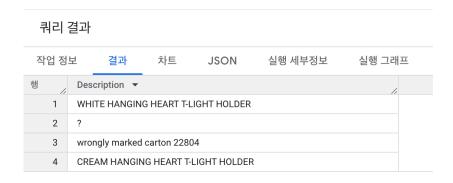
쿼리 결과

작업 정	보 결과	차트	JSC	NC	실행 세부정보	
행 //	column_name	~	//	missin	g_percentage	
1	CustomerID				24.93	
2	UnitPrice				0.0	
3	Description				0.27	
4	InvoiceNo				0.0	
5	Country				0.0	
6	InvoiceDate				0.0	
7	Quantity				0.0	
8	StockCode				0.0	

결측치 처리 전략

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

SELECT Description
FROM `vital-program-456100-p3.modulabs_project.data`
WHERE StockCode = '85123A'
GROUP BY Description



결측치 처리

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

DELETE FROM `vital-program-456100-p3.modulabs_project.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

```
WITH frequency AS (

SELECT *,

COUNT(*) AS frequency

FROM `vital-program-456100-p3.modulabs_project.data`

GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
)

SELECT *

FROM frequency
WHERE frequency.frequency > 1
```



중복값 처리

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



11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

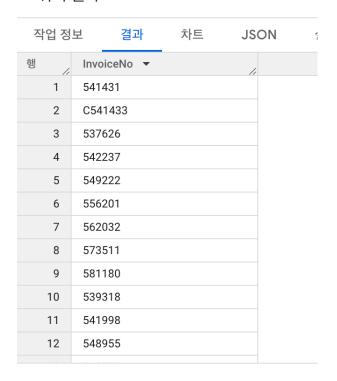
SELECT InvoiceNo, COUNT(InvoiceNo) AS unique_InvoiceNo FROM `vital-program-456100-p3.modulabs_project.data` GROUP BY InvoiceNo

쿼리 결과			② 결과 저장 ▼			
작업 정	보 결과	차트	JS	NC	실행 세	부정보
행	InvoiceNo ▼		//	unique_	InvoiceN	0 7
1	541431					1
2	C541433					1
3	537626				3	31
4	542237				2	29
5	549222				2	24
6	556201				1	8
7	562032				2	22
8	573511				4	17
9	581180				1	1
10	539318				1	17
11	541998					6
12	548955					5
гинлі						

• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

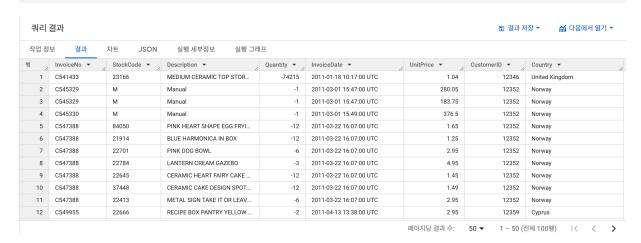
SELECT InvoiceNo FROM `vital-program-456100-p3.modulabs_project.data` GROUP BY InvoiceNo LIMIT 100

쿼리 결과



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

SELECT *
FROM `vital-program-456100-p3.modulabs_project.data`
WHERE InvoiceNo LIKE "C%"
LIMIT 100



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

SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(InvoiceNo) * 100, 1) FROM `vital-program-456100-p3.modulabs_project.data`



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT DISTINCT StockCode FROM `vital-program-456100-p3.modulabs_project.data`

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

SELECT StockCode, COUNT(*) AS sell_cnt FROM project_name.modulabs_project.data GROUP BY StockCode ORDER BY sell_cnt DESC LIMIT 10;



- 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 `vital-program-456100-p3.modulabs_project.data`
)
WHERE number_count <= 1
```

쿼리 :	결과		᠍ 결과 저장 ▼
작업 정	보 결과 차트 JS	ON 실행 세부정	보 실행 그래프
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 number_count <= 1 THEN 1 ELSE 0 END) / COUNT(StockCode) * 100, 2)
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM `vital-program-456100-p3.modulabs_project.data`
)
```



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `vital-program-456100-p3.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 `vital-program-456100-p3.modulabs_project.data`
)
```



Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT Description, COUNT(*) AS description_cnt FROM `vital-program-456100-p3.modulabs_project.data` GROUP BY 1 ORDER BY 2 DESC LIMIT 30



• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE FROM `vital-program-456100-p3.modulabs_project.data`
WHERE
LENGTH(REGEXP_REPLACE(Description, r'[A-Z]', '')) - LENGTH(REGEXP_REPLACE(Description, r'[a-z]', '')) > 0



• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE `vital-program-456100-p3.modulabs_project.data` AS SELECT

* EXCEPT (Description),
UPPER(Description) AS Description
FROM `vital-program-456100-p3.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 `vital-program-456100-p3.modulabs_project.data`



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

SELECT COUNT(Quantity) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS av g_quantity
FROM `vital-program-456100-p3.modulabs_project.data`
WHERE UnitPrice = 0



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE `vital-program-456100-p3.modulabs_project.data` AS SELECT * FROM `vital-program-456100-p3.modulabs_project.data` WHERE UnitPrice != 0.0



11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay, * FROM `vital-program-456100-p3.modulabs_project.data`



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
FROM `vital-program-456100-p3.modulabs_project.data`

-- 예시에서
-- # [[YOUR QUERY]] AS InvoiceDay,
-- *
-- 이 두 줄을 어떻게 살려야 할 지 모르겠어요;
```



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID

쿼리	결과				᠍ 결과 저장 ▼	~
작업 정	보 결과 치	l트 JSON	실행 세부정보	실행 그라	<u> </u> 프	
행	CustomerID ▼	InvoiceDay ▼				
1	12346	2011-01-18				
2	12347	2011-12-07				
3	12348	2011-09-25				
4	12349	2011-11-21				
5	12350	2011-02-02				
6	12352	2011-11-03				
7	12353	2011-05-19				
8	12354	2011-04-21				
9	12355	2011-05-09				
10	12356	2011-11-17				
11	12357	2011-11-06				
12	12358	2011-12-08				
			페이지당 결과 수:	50 ▼	1 - 50 (전체 4362	행)

• 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 `vital-program-456100-p3.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 `vital-program-456100-p3.modulabs_project.user_r` AS

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

FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay

FROM `vital-program-456100-p3.modulabs_project.data`

GROUP BY CustomerID
);
```

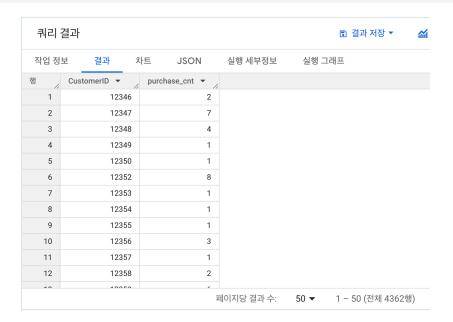


Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
```

FROM `vital-program-456100-p3.modulabs_project.data` GROUP BY CustomerID



• 각 고객 별로 구매한 아이템의 총 수량 더하기

WITH purchase_cnt AS (

SELECT CustomerID, SUM(Quantity) AS item_cnt FROM `vital-program-456100-p3.modulabs_project.data` GROUP BY CustomerID



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

CREATE OR REPLACE TABLE `vital-program-456100-p3.modulabs_project.user_rf` AS
-- (1) 전체 거래 건수 계산

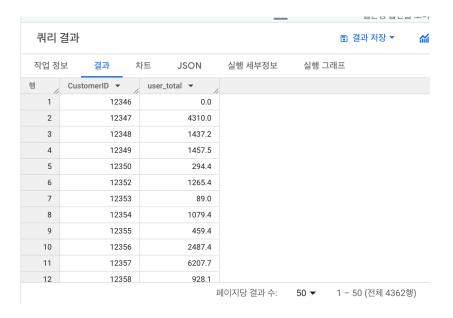
```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT CustomerID,
SUM(Quantity) AS item_cnt
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
)
-- 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 `vital-program-456100-p3.modulabs_project.user_r` AS ur
ON pc.CustomerID = ur.CustomerID;
```



Monetary

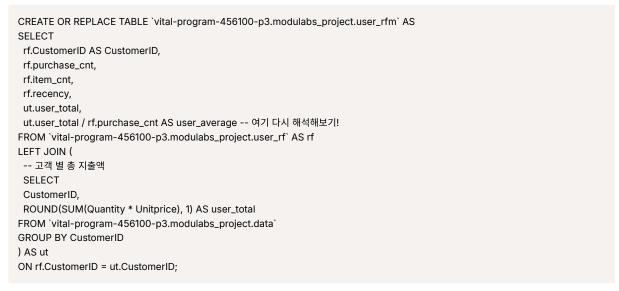
• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(Quantity * Unitprice), 1) AS user_total
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
```



• 고객별 평균 거래 금액 계산

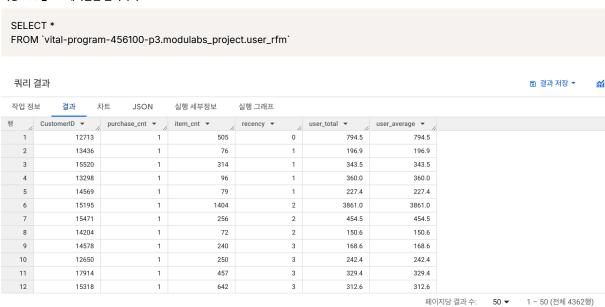
○ 고객별 평균 거래 금액을 구하기 위해 1) data 테이블을 user_rf 테이블과 조인(LEFT JOIN) 한 후, 2) purchase_cnt 로 나누어서 3) user_rfm 테이블로 저장하기





RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기



11-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

1) 고객 별로 구매한 상품들의 고유한 수를 계산하기
 2)

user_rfm 테이블과 결과를 합치기

31

user_data 라는 이름의 테이블에 저장하기

```
CREATE OR REPLACE TABLE vital-program-456100-p3.modulabs_project.user_data AS
WITH unique_products AS (
SELECT
CustomerID,
COUNT(DISTINCT StockCode) AS unique_products
FROM vital-program-456100-p3.modulabs_project.data
GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM vital-program-456100-p3.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 균 구매 소요 일수를 계산하고, 그 결과를 user_data 에 통합

```
CREATE OR REPLACE TABLE vital-program-456100-p3.modulabs_project.user_data AS
WITH purchase_intervals AS (
 -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
 SELECT
 CustomerID,
  CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
 -- (1) 구매와 구매 사이에 소요된 일수
  SELECT
   CustomerID,
   DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
   vital-program-456100-p3.modulabs_projectt.data
  WHERE CustomerID IS NOT NULL
 GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM vital-program-456100-p3.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 vital-program-456100-p3.modulabs_project.user_data AS

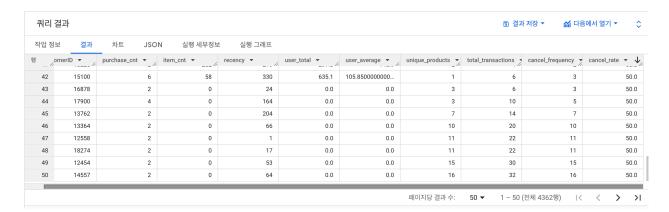
WITH TransactionInfo AS (
SELECT
CustomerID,
COUNT(InvoiceNo) AS total_transactions,
SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
FROM vital-program-456100-p3.modulabs_project.data
GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(cancel_frequency / total_transactions * 100, 1) AS cancel_rate
FROM 'vital-program-456100-p3.modulabs_project.user_data' AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID



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

SELECT *
FROM vital-program-456100-p3.modulabs_project.user_data



회고

[회고 내용을 작성해주세요]

Keep: 손으로 직접 최대한 치면서 하려고 노력했다. 포기하지 않고 열심히 했다.

Problem : 가이드에서 몇 가지 해석이 어려운 것들이 아직 남아 있다. (문법을 몰라서 그런 거 말고 해석) 백업을 안 했는데 다행히 자료 날아간 일은 없지만 백 업 빠뜨리지 않을 것!

Try : 분석 전략을 가질 때 너무 돌아가지 않도록 조금 심플하게 생각해보는 것도 필요!