고객을 세그먼테이션하자 [프로젝트]

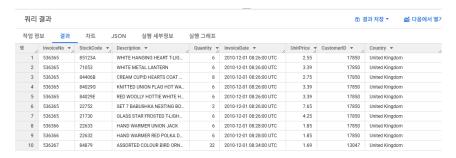
11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
select * from modulabs_project.data
limit 10;
```

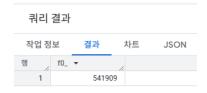
[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
select count(*) from modulabs_project.data
;
```

[결과 이미지를 넣어주세요]



데이터 수 세기

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

```
# [[YOUR QUERY]]
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 modulabs_project.data;

[결과 이미지를 넣어주세요]



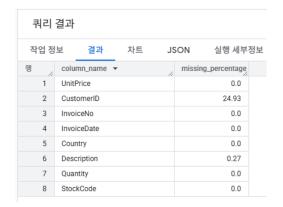
11-4. 데이터 전처리 방법(1): 결측치 제거

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

```
# [[YOUR QUERY]]
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM 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 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 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 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 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 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 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 modulabs_project.data
```

[결과 이미지를 넣어주세요]

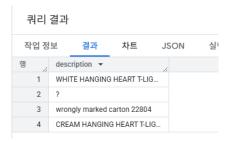


결측치 처리 전략

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

select DISTINCT(description) from modulabs_project.data where StockCode = '85123A';

[결과 이미지를 넣어주세요]



결측치 처리

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

DELETE FROM modulabs_project.data where customerID is null;

[결과 이미지를 넣어주세요]



11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

• 중복된 행의 수를 세어보기

 \circ 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

SELECT COUNT(*) AS duplicate_row_count
FROM (
SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, COUNT(*) AS cnt
FROM modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING COUNT(*) > 1
) AS duplicates;

[결과 이미지를 넣어주세요]



중복값 처리

• 중복값을 제거하는 쿼리문 작성하기

[[YOUR QUERY]];

○ CREATE OR REPLACE TABLE 구문을 활용하여 모든 컬럼(*)을 DISTINCT 한 데이터로 업데이트

select count(*)
from modulabs_project.data;



11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

[[YOUR QUERY]]
select count(distinct(invoiceNo))
from modulabs_project.data;

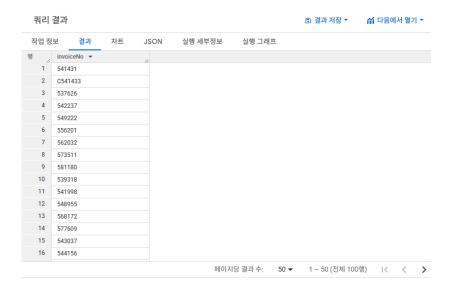
[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

[[YOUR QUERY]] select distinct(invoiceNo) from modulabs_project.data limit 100;

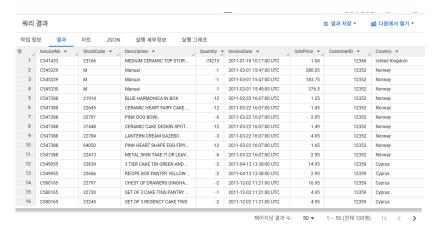
[결과 이미지를 넣어주세요]



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

select *
from `modulabs_project.data`
where invoiceno like 'C%'
LIMIT 100;

[결과 이미지를 넣어주세요]



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

select ROUND(SUM(CASE WHEN quantity <0 THEN 1 ELSE 0 END)/COUNT(*) * 100, 1) from `modulabs_project.data`

[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

[[YOUR QUERY]]
select DISTINCT(StockCode)
from `modulabs_project.data`;

[결과 이미지를 넣어주세요]

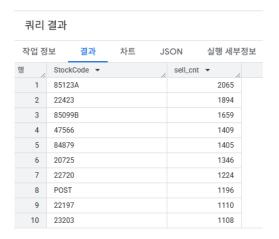


• 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기

。 상위 10개의 제품들을 출력하기

SELECT StockCode, COUNT(*) AS sell_cnt FROM `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]', '')) AS number_count

FROM `modulabs_project.data`
)

WHERE number_count = 0 or number_count = 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과				
작업 정	!보 결과	차트	JSO	N 실행 세부정보
행	StockCode ▼		п	number_count -
1	POST			0
2	М			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(COUNTIF(number_count = 0 OR number_count = 1) / COUNT(*) * 100, 2) as percentage from (

SELECT *,

LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count FROM `modulabs_project.data`
);
```

[결과 이미지를 넣어주세요]



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `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 `modulabs_project.data`
) sub

WHERE number_count = 0 or number_count = 1
);
```

[결과 이미지를 넣어주세요]



Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

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

[결과 이미지를 넣어주세요]

쿼리 결과



• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE from `modulabs_project.data` where Description in ('High Resolution Image','Next Day Carriage');

[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`;

[결과 이미지를 넣어주세요]



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 `modulabs_project.data`;

[결과 이미지를 넣어주세요]



• 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 avg_quant FROM `modulabs_project.data`
WHERE unitprice = 0.0;

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE `modulabs_project.data` AS SELECT * from `modulabs_project.data` where unitprice != 0.0;

[결과 이미지를 넣어주세요]



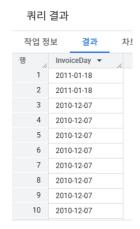
11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay from `modulabs_project.data`

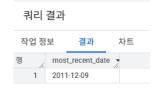
[결과 이미지를 넣어주세요]



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT max(date(InvoiceDate)) as most_recent_date from `modulabs_project.data`

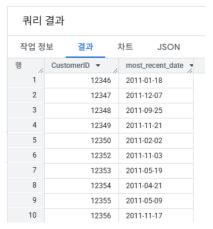
[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

SELECT
CustomerID,
max(date(InvoiceDate)) as most_recent_date
FROM `modulabs_project.data`
group by customerID;

[결과 이미지를 넣어주세요]



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

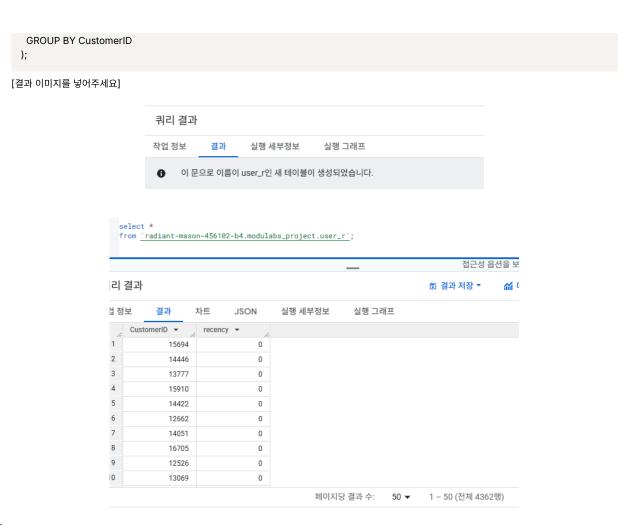
```
SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM project_name.modulabs_project.data
GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE `radiant-mason-456102-b4.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 `radiant-mason-456102-b4.modulabs_project.data`
```



Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
count(DISTINCT InvoiceNo) AS purchase_cnt
FROM `radiant-mason-456102-b4.modulabs_project.data`
group by CustomerID;
```

[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
sum(Quantity) AS item_cnt
FROM `radiant-mason-456102-b4.modulabs_project.data`
group by CustomerID;
```

[결과 이미지를 넣어주세요]

결과 차트 작업 정보 **JSON** CustomerID ▼

쿼리 결과

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

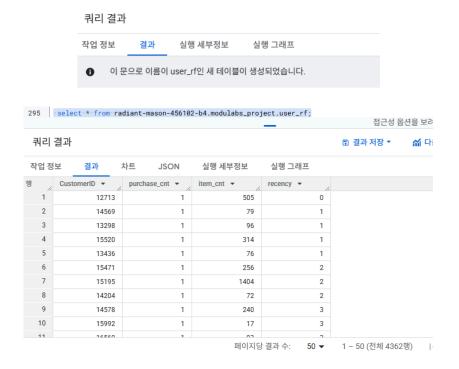
```
CREATE OR REPLACE TABLE `radiant-mason-456102-b4.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
CustomerID,
count(DISTINCT InvoiceNo) AS purchase_cnt
FROM `radiant-mason-456102-b4.modulabs_project.data`
group by CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
```

```
SELECT
 CustomerID,
 sum(Quantity) AS item_cnt
 FROM `radiant-mason-456102-b4.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 'radiant-mason-456102-b4.modulabs_project.user_r' AS ur
ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

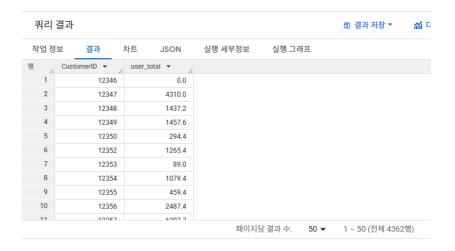


Monetary

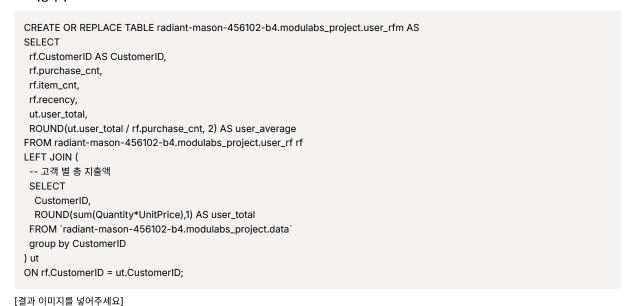
• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(sum(Quantity*UnitPrice),1) AS user_total
FROM `radiant-mason-456102-b4.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



쿼리 결과

결과

작업 정보

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

실행 세부정보

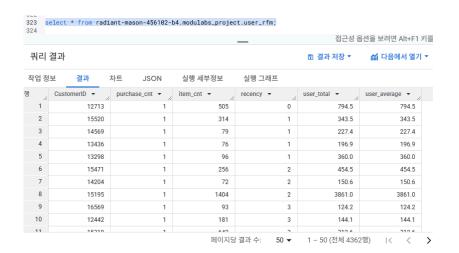
실행 그래프

RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

[[YOUR QUERY]]; select * from radiant-mason-456102-b4.modulabs_project.user_rfm;

[결과 이미지를 넣어주세요]



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

[결과 이미지를 넣어주세요]





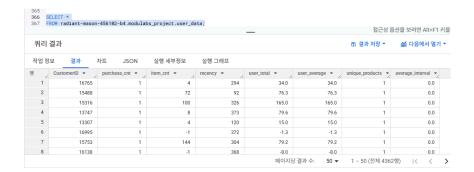
2. 평균 구매 주기

- 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 radiant-mason-456102-b4.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
  FROM (
   -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
    CustomerID,
    DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
    radiant-mason-456102-b4.modulabs_project.data
   WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
 SELECT u.*, pi.* EXCEPT (CustomerID)
 FROM radiant-mason-456102-b4.modulabs_project.user_data AS u
 LEFT JOIN purchase_intervals AS pi
 ON u.CustomerID = pi.CustomerID;
[결과 이미지를 넣어주세요]
```

작업 정보 <u>결과</u> 실행 세부정보 실행 그래프

① 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 radiant-mason-456102-b4.modulabs_project.user_data AS

WITH TransactionInfo AS (
SELECT
CustomerID,
COUNT(*) AS total_transactions,
COUNTIF(LEFT(InvoiceNo, 1) = 'C') AS cancel_frequency
FROM radiant-mason-456102-b4.modulabs_project.data
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / NULLIF(t.total_transactions, 0), 2) AS cancel_rate
FROM `radiant-mason-456102-b4.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

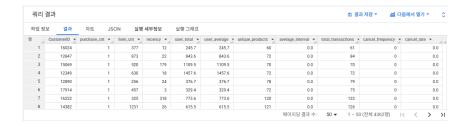
[결과 이미지를 넣어주세요]



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

```
# [[YOUR QUERY]];
SELECT *
FROM radiant-mason-456102-b4.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

Keep : 팀원 분들의 어려운 점을 물어보고 도움을 줄 수 있는 만큼 도와드렸다. 과정을 시간안에 마칠 수 있어서 다행이었다.

Problem : 결과값이 다르게 나오는 이유를 못 찾아서 헤맸었는데, ChatGPT 의 도움으로 문법 오류를 찾아낼 수 있었다.

Try : 문법이나 용어, 개념을 정리해야겠다.