고객을 세그먼테이션하자 [이환철]

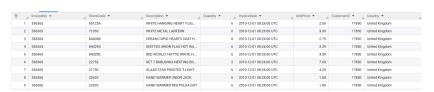
11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM `modulabs_project.data`
LIMIT 10
```

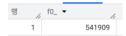
[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT count(*)
FROM `modulabs_project.data`
```

[결과 이미지를 넣어주세요]



데이터 수 세기

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

```
SELECT
count(InvoiceNo) COUNT_InvoiceNo,
count(StockCode) COUNT_StockCode,
count(Description) COUNT_Description,
count(Quantity) Count_Quantity,
count(InvoiceDate) Count_InvoiceDate,
count(UnitPrice) Count_UnitPrice,
count(CustomerID) Count_CustomerID,
count(Country) Count_Country
FROM
`modulabs_project.data`
```

[결과 이미지를 넣어주세요]



11-4. 데이터 전처리 방법(1): 결측치 제거

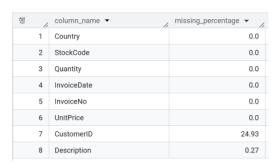
컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계산
 - 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

```
SELECT column_name,
   ROUND((total - column_value) / total * 100, 2) as missing_percentage
FROM
(

SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM `modulabs_project.da'
SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM `modulabs_project.c
SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM `modulabs_project.
SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM `modulabs_project.data'
SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM `modulabs_project.data'
SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM `modulabs_project.data'
SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM `modulabs_project.data'
SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM `modulabs_project.data'
SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total 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
Description IS NULL OR CustomerID IS NULL;
```

[결과 이미지를 넣어주세요]



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
COUNT(*) as count_duplicated
FROM (
SELECT *, COUNT(*) as count
FROM `modulabs_project.data`
GROUP BY
InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING COUNT(*) > 1
}
```

[결과 이미지를 넣어주세요]



중복값 처리

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

```
CREATE OR REPLACE TABLE 'modulabs_project.data' AS
SELECT DISTINCT *
FROM 'modulabs_project.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 `modulabs_project.data`

[결과 이미지를 넣어주세요]

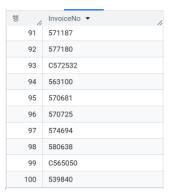


• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT
DISTINCT InvoiceNo
FROM `modulabs_project.data`
LIMIT 100

[결과 이미지를 넣어주세요]



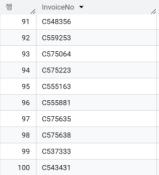


• InvoiceNo 가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)

SELECT *
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]]
LIMIT 100;

[결과 이미지를 넣어주세요]





• 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) FROM `modulabs_project.data`;

[결과 이미지를 넣어주세요]



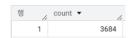
StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT

COUNT(DISTINCT StockCode) count FROM `modulabs_project.data`

[결과 이미지를 넣어주세요]



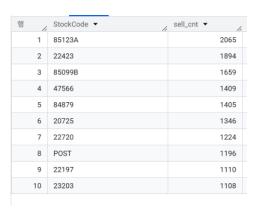
• 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기

。 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 값들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT StockCode, number_count

FROM (

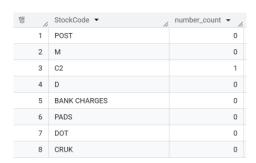
SELECT StockCode,

LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count

FROM `modulabs_project.data`
)

WHERE number_count <= 1
```

[결과 이미지를 넣어주세요]



- 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)

SELECT

ROUND(SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1 THEN 1 ELSE 0 E FROM `modulabs_project.data`;

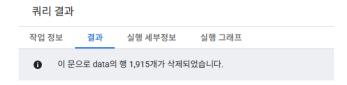
[결과 이미지를 넣어주세요]



• 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`
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1
)
);

[결과 이미지를 넣어주세요]
```

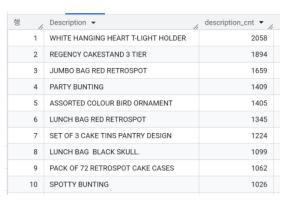


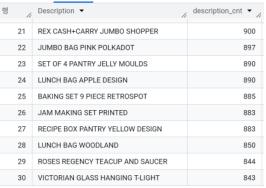
Description 살펴보기

• 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]





• 서비스 관련 정보를 포함하는 행들을 제거하기



• 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

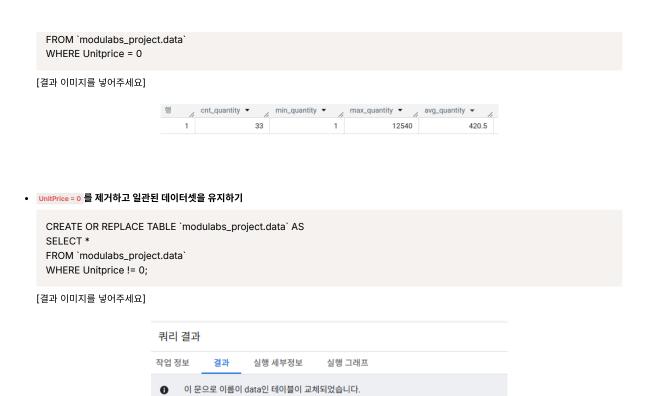
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, round(AVG(UnitPrice),1) AS avg_price FROM `modulabs_project.data`;

[결과 이미지를 넣어주세요]



• 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),1) AS avg_quantity



11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDATE) AS InvoiceDay FROM `modulabs_project.data`

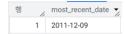
[결과 이미지를 넣어주세요]



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT DATE(MAX(InvoiceDATE)) AS most_recent_date FROM `modulabs_project.data`

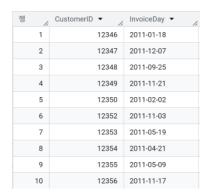
[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
DATE(MAX(InvoiceDATE)) AS InvoiceDay
FROM `modulabs_project.data`
GROUP BY CustomerID
```

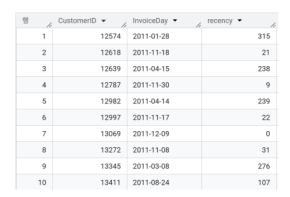
[결과 이미지를 넣어주세요]

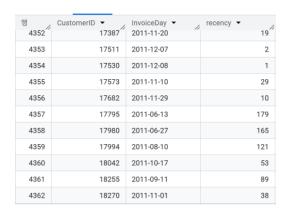


• 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산하기

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

[결과 이미지를 넣어주세요]

라리 결과
작업정보 결과 실행세부정보 실행그래프

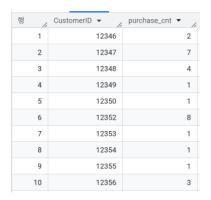
① 이 문으로 아름이 user_r인 새 테이블이 생성되었습니다.
```

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
count(DISTINCT InvoiceNo) AS purchase_cnt
FROM `modulabs_project.data`
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

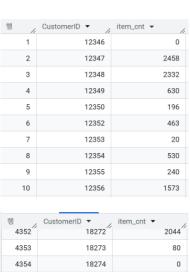


행 //	CustomerID	*	purchase_cnt ▼	-
4352		18272		7"
4353		18273		3
4354		18274		2
4355		18276		3
4356		18277		2
4357		18278		1
4358		18280		1
4359		18281		1
4360		18282		3
4361		18283		16
4362		18287		3

• 각 고객 별로 구매한 아이템의 총 수량 더하기

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

[결과 이미지를 넣어주세요]



행 //	CustomerID	· //	item_cnt ▼	/
4352		18272		2044
4353		18273		80
4354		18274		0
4355		18276		184
4356		18277		67
4357		18278		66
4358		18280		45
4359		18281		54
4360		18282		98
4361		18283		1355
4362		18287		1586

• 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

WITH purchase_cnt AS (

SELECT

CustomerID,

count(DISTINCT InvoiceNo) AS purchase_cnt

```
FROM 'modulabs_project.data'
  GROUP BY CustomerID
 ),
 item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM 'modulabs_project.data'
  GROUP BY CustomerID
 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;
[결과 이미지를 넣어주세요]
```

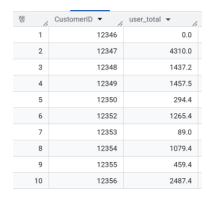


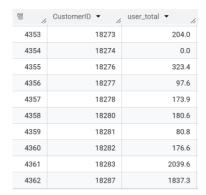
Monetary

• 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
```

[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

```
CREATE OR REPLACE TABLE 'modulabs_project.user_rfm' AS
  SELECT
   rf.CustomerID AS CustomerID,
   rf.purchase_cnt,
   rf.item_cnt,
   rf.recency,
   ROUND(ut.user_total, 0) AS user_total,
   ROUND(ut.user_total / rf.purchase_cnt, 0) AS user_average
  FROM `modulabs_project.user_rf` rf
  LEFT JOIN (
   SELECT
    CustomerID,
    SUM(Quantity*UnitPrice) AS user_total
   FROM 'modulabs_project.data'
   GROUP BY CustomerID
  ) ut
  ON rf.CustomerID = ut.CustomerID;
[결과 이미지를 넣어주세요]
```



RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM `modulabs_project.user_rfm`
```

[결과 이미지를 넣어주세요]

행 //	CustomerID ▼	purchase_cnt ▼ //	item_cnt ▼	recency ▼	user_total ▼	user_average ▼ //
1	12713	1	505	0	795.0	795.0
2	14569	1	79	1	227.0	227.0
3	13436	1	76	1	197.0	197.0
4	13298	1	96	1	360.0	360.0
5	15520	1	314	1	344.0	344.0
6	14204	1	72	2	151.0	151.0
7	15471	1	256	2	454.0	454.0
8	15195	1	1404	2	3861.0	3861.0
9	15992	1	17	3	42.0	42.0
10	17914	1	457	3	329.0	329.0
행 #304	CustomerID V	purchase_cnt ▼	item_cnt ▼	recency ▼	user_total ▼	user_average ▼
행 #304	CustomerID ▼	purchase_cnt ▼	item_cnt ▼	recency ▼	user_total ▼	user_average ▼
4353	16029	66	33687	38	60370.0	915.0
4354	14646	73	196556	1	278778.0	3819.0
4355	13408	75	16128	1	27888.0	372.0
4356	12971	88	9204	3	10934.0	124.0
4357	15311	118	37673	0	59284.0	502.0
	13089	118	30742	2	57322.0	486.0
4358				1	11487.0	92.0
4358 4359	14606	125	5932			
	14606 17841	125 169	5932 22613	1	39861.0	236.0
4359				1	39861.0 29820.0	236.0 137.0

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 `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;
```

[결과 이미지를 넣어주세요]

```
    쿼리 결과

    작업 정보
    결과
    실행 세부정보
    실행 그래프

    ⑥ 이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.
```

2. 평균 구매 주기

• 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)

```
○ 평균 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 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_
     `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;
[결과 이미지를 넣어주세요]
```

쿼리 결과					
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 '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 'modulabs_project.data'
GROUP BY CustomerID
SELECT
u.*, t.* EXCEPT(CustomerID),
ROUND(IF(t.total_transactions = 0, 0, t.cancel_frequency / t.total_transactions),2) AS cancel_rate
 --ROUND(t.cancel_frequency / t.total_transactions,2) AS cancel_rate
FROM 'modulabs_project.user_data' AS u
```

LEFT JOIN TransactionInfo AS t ON u.CustomerID = t.CustomerID

[결과 이미지를 넣어주세요]

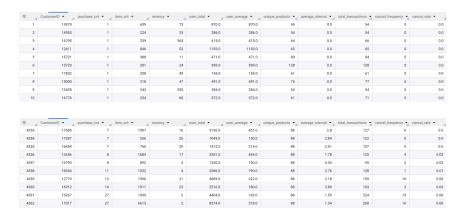


• 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 user_data 를 출력하기

SELECT *

FROM 'modulabs_project.user_data'

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

Keep:

- SQL 쿼리를 계속 작성하면서 작성 과정을 우선 글로 적은 후 쿼리 작성하는 방식으로 SQL 쿼리 논리 흐름을 익히는 연습을 꾸준히 하고 있음
- SQL을 활용한 데이터 전처리 및 분석에 필요한 전체 흐름을 머릿속에 그리며 점점 익숙해지는 과정이라고 느꼈음

Problem:

SQL학습 환경별 SQL이 달라서 그런지 DATE, Timestamp 타입 전처리는 여전히 익숙하지 않네요.

- 학습 환경(Google BigQuery, MySQL, SQLite 등)에 따라 함수나 문법 차이(Date, Timestamp 등)**가 있어 혼란이 다소 있음
- 특히, 날짜/시간 처리 함수의 경우 시행착오가 많았음
- 또한 쿼리 최적화 또는 서브쿼리 구조 등은 아직 익숙하지 않아서 작성에 시간이 다소 소요됨

Try:

- 실무에서 자주 쓰이는 날짜 처리, 윈도우 함수, 집계 쿼리에 대한 유형별 템플릿을 만들어 반복 숙련
- SQLD 자격증을 단기 목표로 삼아 체계적인 SQL 이해와 실력 향상을 꾀할 예정