고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

SELECT *
FROM my-first-project-466601.modulabs_project.data
LIMIT 10

[결과 이미지를 넣어주세요]



• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

SELECT COUNT(*)
FROM my-first-project-466601.modulabs_project.data

[결과 이미지를 넣어주세요]



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 my-first-project-466601.modulabs_project.data]

[결과 이미지를 넣어주세요]



11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

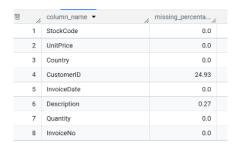
• 각 컬럼 별 누락된 값의 비율을 계산

\circ 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

```
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(IF(InvoiceNo IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
UNION ALL
SELECT
  'StockCode' AS column_name,
  ROUND(SUM(IF(StockCode IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
UNION ALL
SELECT.
  'Description' AS column_name,
  ROUND(SUM(IF(Description IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
UNION ALL
SELECT.
  'Quantity' AS column_name,
  ROUND(SUM(IF(Quantity IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
UNION ALL
SELECT
  'InvoiceDate' AS column_name,
  ROUND(SUM(IF(InvoiceDate IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
UNION ALL
SELECT
  'UnitPrice' AS column_name,
  ROUND(SUM(IF(UnitPrice IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
UNION ALL
SELECT
  'CustomerID' AS column_name,
  ROUND(SUM(IF(CustomerID IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
FROM my-first-project-466601.modulabs_project.data
UNION ALL
SELECT
  'Country' AS column_name,
  ROUND(SUM(IF(Country IS NULL, 1, 0)) / COUNT(*) * 100, 2) AS missing_percentage
```

[결과 이미지를 넣어주세요]

FROM my-first-project-466601.modulabs_project.data



결측치 처리 전략

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

SELECT DISTINCT(Description)
FROM my-first-project-466601.modulabs_project.data
WHERE StockCode = '85123A'

[결과 이미지를 넣어주세요]



결측치 처리

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

DELETE
FROM my-first-project-466601.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
 - \circ 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

```
SELECT COUNT(*)
FROM (
SELECT *
FROM my-first-project-466601.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 my-first-project-466601.modulabs_project.data AS
SELECT DISTINCT *
FROM my-first-project-466601.modulabs_project.data;

SELECT COUNT(*)
FROM my-first-project-466601.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)
FROM my-first-project-466601.modulabs_project.data

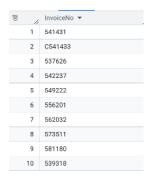
[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo FROM my-first-project-466601.modulabs_project.data LIMIT 100

[결과 이미지를 넣어주세요]



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

SELECT *
FROM my-first-project-466601.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100

[결과 이미지를 넣어주세요]

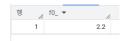


• 구매 건 상태가 Canceled 인 데이터의 비율(%) - 소수점 첫번째 자리까지

SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) * 100 / COUNT(*), 1) FROM my-first-project-466601.modulabs_project.data

SELECT ROUND(SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) * 100 / COUNT(*), 1) FROM my-first-project-466601.modulabs_project.data

[결과 이미지를 넣어주세요]

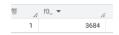


StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

SELECT COUNT(DISTINCT StockCode)
FROM my-first-project-466601.modulabs_project.data

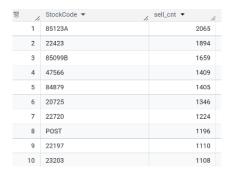
[결과 이미지를 넣어주세요]



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

SELECT StockCode, COUNT(*) AS sell_cnt FROM my-first-project-466601.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 my-first-project-466601.modulabs_project.data
)
WHERE number_count IN (0, 1)
```

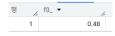
[결과 이미지를 넣어주세요]

행 //	StockCode ▼	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)

```
WITH UNUSUAL_CODE AS(
SELECT DISTINCT StockCode, number_count
FROM (
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM my-first-project-466601.modulabs_project.data
)
WHERE number_count IN (0, 1)
)
SELECT
ROUND(COUNT(CASE WHEN StockCode IN (SELECT StockCode FROM UNUSUAL_CODE) THEN 1 END) * 100 / COUNT(*), 2)
FROM my-first-project-466601.modulabs_project.data
```

[결과 이미지를 넣어주세요]



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM my-first-project-466601.modulabs_project.data

WHERE StockCode IN (

SELECT DISTINCT StockCode

FROM (

SELECT DISTINCT StockCode, number_count

FROM (

SELECT StockCode,

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

FROM my-first-project-466601.modulabs_project.data
)

WHERE number_count IN (0, 1)
```

```
)
)
[결과 이미지를 넣어주세요]
```

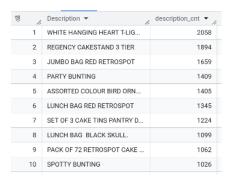
이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT Description, COUNT(*) AS description_cnt FROM my-first-project-466601.modulabs_project.data GROUP BY Description ORDER BY description_cnt DESC LIMIT 30

[결과 이미지를 넣어주세요]



• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE
FROM my-first-project-466601.modulabs_project.data
WHERE
UPPER(Description) IN (UPPER('Next Day Carriage'), UPPER('High Resolution Image'))

[결과 이미지를 넣어주세요]

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE my-first-project-466601.modulabs_project.data AS SELECT

* EXCEPT (Description),

UPPER(Description) AS Description

FROM my-first-project-466601.modulabs_project.data

[결과 이미지를 넣어주세요]

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, AVG(UnitPrice) AS avg_price FROM my-first-project-466601.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 ϵ FROM my-first-project-466601.modulabs_project.data WHERE UnitPrice = 0

[결과 이미지를 넣어주세요]



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

CREATE OR REPLACE TABLE my-first-project-466601.modulabs_project.data AS SELECT *
FROM my-first-project-466601.modulabs_project.data
WHERE UnitPrice != 0

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

SELECT DATE(InvoiceDate) AS InvoiceDay, * FROM my-first-project-466601.modulabs_project.data

[결과 이미지를 넣어주세요]

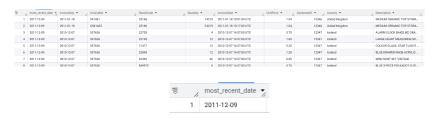


• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT
(SELECT MAX(DATE(InvoiceDate)) FROM my-first-project-466601.modulabs_project.data) AS most_recent_date,
DATE(InvoiceDate) AS InvoiceDay,
*
FROM my-first-project-466601.modulabs_project.data;

SELECT MAX(DATE(InvoiceDate)) AS most_recent_date FROM my-first-project-466601.modulabs_project.data;

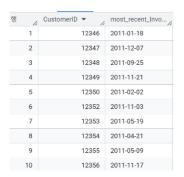
[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS most_recent_InvoiceDay
FROM my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
ORDER 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 my-first-project-466601.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 my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
)
ORDER BY recency;
```

[결과 이미지를 넣어주세요]

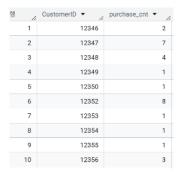


Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
ORDER BY CustomerID;
```

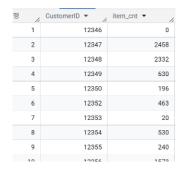
[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

[결과 이미지를 넣어주세요]

```
SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM my-first-project-466601.modulabs_project.data
GROUP BY CustomerID
ORDER BY CustomerID;
```



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

```
CREATE OR REPLACE TABLE my-first-project-466601.modulabs_project.user_rf AS
WITH purchase_cnt AS (
 SELECT
   CustomerID,
   COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY CustomerID
  ORDER BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
   CustomerID,
   SUM(Quantity) AS item_cnt
  FROM my-first-project-466601.modulabs_project.data
  GROUP BY CustomerID
  ORDER 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 my-first-project-466601.modulabs_project.user_r AS ur
 ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

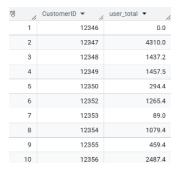
행 //	CustomerID ▼	purchase_cnt ▼ //	item_cnt ▼	recency ▼
1	12713	1	505	0
2	15520	1	314	1
3	14569	1	79	1
4	13436	1	76	1
5	13298	1	96	1
6	14204	1	72	2
7	15471	1	256	2
8	15195	1	1404	2
9	16569	1	93	3
10	16528	1	171	3

Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM my-first-project-466601.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 my-first-project-466601.modulabs_project.user_rfm AS
SELECT
 rf.CustomerID AS CustomerID,
 rf.purchase_cnt,
 rf.item_cnt,
 rf.recency,
 ut.user_total,
 ut.user_total / rf.purchase_cnt AS user_average
FROM my-first-project-466601.modulabs_project.user_rf rf
LEFT JOIN (
 SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
 FROM my-first-project-466601.modulabs_project.data
 GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

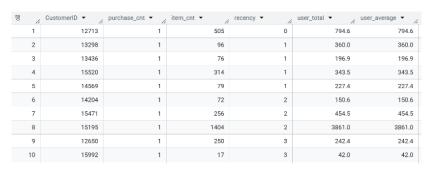


RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

SELECT *
FROM my-first-project-466601.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 my-first-project-466601.modulabs_project.user_data AS

WITH unique_products AS (

SELECT

CustomerID,

COUNT(DISTINCT StockCode) AS unique_products

FROM my-first-project-466601.modulabs_project.data

GROUP BY CustomerID
)

SELECT ur.*, up.* EXCEPT (CustomerID)

FROM my-first-project-466601.modulabs_project.user_rfm AS ur

JOIN unique_products AS up

ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

행 //	CustomerID ▼	purchase_cnt ▼ //	item_cnt ▼	recency ▼	user_total ▼ //	user_average ▼ //	unique_products ▼//
1	12713	1	505	0	794.6	794.6	37
2	15520	1	314	1	343.5	343.5	18
3	13298	1	96	1	360.0	360.0	2
4	13436	1	76	1	196.9	196.9	12
5	14569	1	79	1	227.4	227.4	10
6	14204	1	72	2	150.6	150.6	36
7	15471	1	256	2	454.5	454.5	67
8	15195	1	1404	2	3861.0	3861.0	1
9	14578	1	240	3	168.6	168.6	24
10	12478	1	233	3	546.0	546.0	35

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 project_name.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_
   project_name.modulabs_project.data
 WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

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 my-first-project-466601.modulabs_project.user_data AS

WITH TransactionInfo AS (

SELECT

CustomerID,

COUNT(*) AS total_transactions,

COUNT(CASE WHEN Quantity < 0 THEN 1 END) AS cancel_frequency

FROM my-first-project-466601.modulabs_project.data

GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate

FROM my-first-project-466601.modulabs_project.user_data AS u

LEFT JOIN TransactionInfo AS t

ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

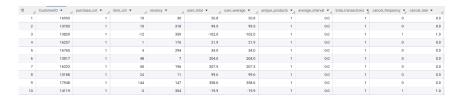
8 /	CustomeriD ▼	purchase_cnt *	item_cnt •	recency •	user_total ▼	user_average ▼	unique_products •	average_interval 🕶 //	total_transactions >	cancel_frequency -	canceLrate ▼
- 1	16953	1	10	30	20.8	20.8	1	0.0	1	0	0.0
2	13703	1	10	318	99.5	99.5	1	0.0	1	0	0.0
3	13829	1	-12	359	-102.0	-102.0	1	0.0	1	1	1.0
4	16257	1	1	176	21.9	21.9	1	0.0	1	0	0.0
5	16765	1	4	294	34.0	34.0	1	0.0	1	0	0.0
6	13017	1	48	7	204.0	204.0	1	0.0	1	0	0.0
7	16323	1	50	196	207.5	207.5	1	0.0	1	0	0.0
8	13188	1	24	11	99.6	99.6	1	0.0	1	0	0.0
9	17948	1	144	147	358.6	358.6	1	0.0	1	0	0.0
10	14119	1	-2	354	-19.9	-19.9	1	0.0	1	1	1.0

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

SELECT *

FROM my-first-project-466601.modulabs_project.user_data

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

Keep : 다양한 USER FEATURE를 통해서 유저 분석이 가능하도록 데이터를 전처리 할 수 있다. Problem : 실수로 조건을 잘못 입력해서 CREAT OR REPLACE TABLE 하면 고생한다..ㅠㅠ

Try : 다양한 소스들을 통해 SQL 쿼리 연습해야겠다.