

고객을 세그먼테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
SELECT *
FROM modulabs_project.data
LIMIT 10;
```

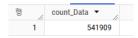
[결과 이미지를 넣어주세요]

ğ	InvoiceNo ▼	StockCode ▼	Description ▼	Quantity -	InvoiceDate ▼	UnitPrice ▼	CustomerID ▼	Country -
1	536414	22139	nuit	56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	37509	nuili	1	2010-12-01 14:33:00 UTC	0.0	Run	United Kingdom
5	536549	85226A	nuil	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
6	536550	85044	null'	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
7	536552	20950	nuil	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
9	536554	84670	nuit	23	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	0.0	null	United Kingdom

• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
SELECT COUNT(*) AS count_Data
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]



데이터 수 세기

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

```
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

5-4. 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

```
# [[YOUR QUERY]]
SELECT
   'InvoiceNo' AS column_name,
   ROUND(COUNTIF(InvoiceNo IS NULL) / COUNT(*) * 100, 2) AS missing_percentage
FROM modulabs_project.data
UNION ALL
```

```
SELECT

'StockCode' AS column_name,

ROUND(COUNTIF(StockCode IS NULL) / COUNT(*) * 100, 2) AS missing_percentage

FROM modulabs_project.data

UNION ALL

SELECT

'Description' AS column_name,

ROUND(COUNTIF(Description IS NULL) / COUNT(*) * 100, 2) AS missing_percentage

FROM modulabs_project.data
```

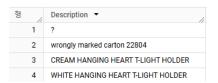
행 //	column_name ▼	missing_percentage
1	StockCode	0.0
2	InvoiceNo	0.0
3	Description	0.27

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM modulabs_project.data
WHERE StockCode = '85123A'
AND Description IS NOT NULL;
```

[결과 이미지를 넣어주세요]



결측치 처리

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

```
DELETE FROM modulabs_project.data
WHERE InvoiceNo IS NULL
OR StockCode IS NULL
OR Description IS NULL
OR Quantity IS NULL
OR InvoiceDate IS NULL
OR UnitPrice IS NULL
OR CustomerID IS NULL
OR Country IS NULL;
```

[결과 이미지를 넣어주세요]

● 이 문으로 data의 행 0개가 삭제되었습니다.

5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- 중복된 행의 수를 세어보기
 - $_{\circ}~$ 8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기

```
WITH group_data AS (
 SELECT
   InvoiceNo,
   StockCode,
   Description,
    Quantity,
   InvoiceDate,
   UnitPrice,
   CustomerID,
   Country,
   COUNT(*) AS row_count
 FROM modulabs_project.data
  GROUP BY
    InvoiceNo,
    StockCode,
   Description,
   Quantity,
   InvoiceDate,
   UnitPrice,
    CustomerID,
    Country
SELECT COUNT(*)
FROM group_data
WHERE row_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 modulabs_project.data AS
SELECT DISTINCT *
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

● 이 문으로 이름이 data인 테이블이 교체되었습니다.

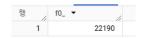
5-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

• 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM modulabs_project.data;
```

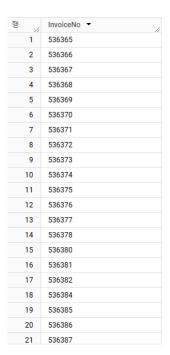
[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

SELECT DISTINCT InvoiceNo FROM modulabs_project.data ORDER BY InvoiceNo LIMIT 100;

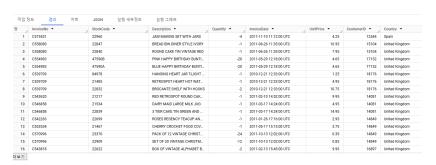
[결과 이미지를 넣어주세요]



• 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]

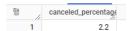


• 구매 건 상태가 Canceled 인 데이터의 비율(%) - 소수점 첫번째 자리까지

SELECT

ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 1) AS canceled_perc FROM modulabs_project.data

[결과 이미지를 넣어주세요]

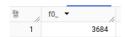


StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

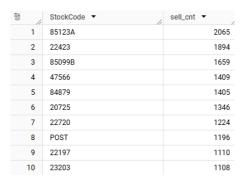


• 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기

。 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 문자열 내 숫자의 길이를 구해보기

```
WITH UniqueStockCodes AS (
    SELECT DISTINCT StockCode
    FROM project_name.modulabs_project.data
)

SELECT
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
    COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC;
```

[결과 이미지를 넣어주세요]

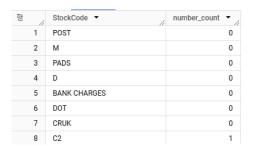
작업 정	보 결과		차트	JSON
행 //	number_count	- //	stock_cnt	· /
1		5		3676
2		0		7
3		1		1

• 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 <= 1;</pre>
```

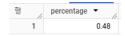
[결과 이미지를 넣어주세요]



- 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)

```
WITH char_count_data AS (
    SELECT
    StockCode,
    LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS char_length,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM modulabs_project.data
)
SELECT
    ROUND(COUNTIF(number_count <= 1) * 100.0 / COUNT(*), 2) AS percentage
FROM char_count_data;</pre>
```

[결과 이미지를 넣어주세요]



• 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM project_name.modulabs_project.data
WHERE StockCode IN (
SELECT DISTINCT StockCode
FROM (
SELECT StockCode
FROM modulabs_project.data
```

```
WHERE StockCode IN ('BANK CHARGES', 'POST', 'D', 'M', 'CRUK')
   OR StockCode LIKE 'GIFT%'
   OR Quantity <= 0
   OR UnitPrice <= 0
)
);</pre>
```

이 문으로 data의 행 0개가 삭제되었습니다.

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]

행	Description ▼	description_cnt ▼
1	FELTCRAFT PRINCESS CHARL	408
2	HAND WARMER RED LOVE HE	386
3	POTTERING IN THE SHED MET	377
4	PACK OF 12 HEARTS DESIGN T	356
5	SET/6 RED SPOTTY PAPER PL	328
6	SET OF 60 VINTAGE LEAF CAK	312
7	WASHROOM METAL SIGN	296
8	SET/6 RED SPOTTY PAPER CU	292
9	CHILDRENS APRON APPLES D	279
10	VINTAGE SNAKES & LADDERS	268
11	JAZZ HEARTS PURSE NOTEBO	258
12	ROBOT BIRTHDAY CARD	258
13	PICTURE DOMINOES	249
14	SET OF 3 CAKE TINS SKETCHB	247
15	SWALLOWS GREETING CARD	243
16	TRADITIONAL PICK UP STICKS	241
17	FANCY FONTS BIRTHDAY WRAP	230
18	HEART IVORY TRELLIS SMALL	226
19	TRADITIONAL NAUGHTS & CR	221
더보기		

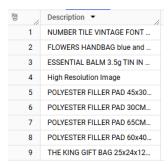
• 대소문자가 혼합된 Description이 있는지 확인하기

```
SELECT DISTINCT Description

FROM project_name.modulabs_project.data

WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

[결과 이미지를 넣어주세요]



• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM modulabs_project.data
WHERE

LOWER(Description) LIKE '%service%'

OR LOWER(Description) LIKE '%repair%'

OR LOWER(Description) LIKE '%postage%'

OR LOWER(Description) LIKE '%delivery%'

OR LOWER(Description) LIKE '%fee%'

OR LOWER(Description) LIKE '%charge%';
```

[결과 이미지를 넣어주세요]

● 이 문으로 data의 행 497개가 삭제되었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS

SELECT

* EXCEPT (Description),

UPPER(Description) AS Description

FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

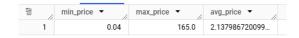
● 이 문으로 이름이 data인 테이블이 교체되었습니다.

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 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,

AVG(quantity) AS avg_quantity

FROM modulabs_project.data

WHERE UnitPrice = 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;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 테이블이 교체되었습니다.

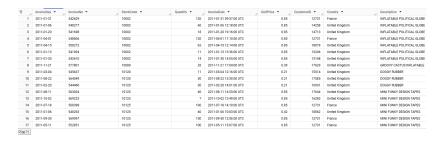
5-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```

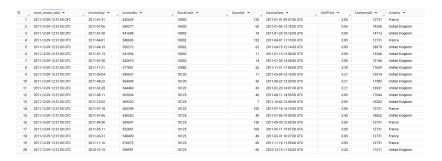
[결과 이미지를 넣어주세요]



• 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
(SELECT MAX(InvoiceDate) FROM modulabs_project.data) AS most_recent_date,
DATE(InvoiceDate) AS InvoiceDay,
```

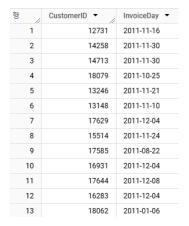
```
*
FROM modulabs_project.data;
```



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

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

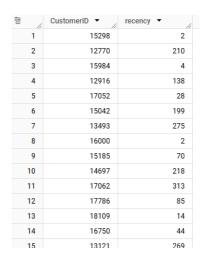
[결과 이미지를 넣어주세요]



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

```
SELECT
  CustomerID,
  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 project_name.modulabs_project.user_r AS

SELECT

CustomerID,

MAX(DATE(InvoiceDate)) AS last_purchase_date,

COUNT(DISTINCT InvoiceNo) AS frequency,

SUM(Quantity * UnitPrice) AS monetary,

EXTRACT(DAY FROM (SELECT MAX(DATE(InvoiceDate)) FROM midyear-module-447402-s8.modulabs_project.dat

FROM midyear-module-447402-s8.modulabs_project.data

WHERE Quantity > 0 AND UnitPrice > 0

GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

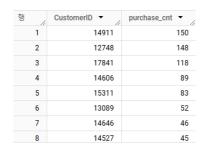
● 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM project_name.modulabs_project.data
GROUP BY CustomerID
ORDER BY purchase_cnt DESC;
```

[결과 이미지를 넣어주세요]



• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
SUM(Quantity) AS item_cnt
FROM modulabs_project.data
WHERE Quantity > 0
GROUP BY CustomerID
ORDER BY item_cnt DESC;
```

[결과 이미지를 넣어주세요]

행	CustomerID ▼	item_cnt ▼
1	14298	22314
2	14646	20710
3	13694	18115
4	14911	13543
5	17511	13273
6	18102	11164
7	12415	8116
8	16308	8000
9	17450	7326
10	14156	6394

• 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
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
 SELECT
   CustomerID,
   COUNT(DISTINCT InvoiceNo) AS purchase_cnt
 FROM `modulabs_project.data`
 GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
 SELECT
   CustomerID,
   SUM(Quantity) AS item_cnt
 FROM `modulabs_project.data`
 WHERE Quantity > 0
 GROUP BY CustomerID
-- 기존의 user_r에 (1)과 (2)를 통합
 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 project_name.modulabs_project.user_r AS ur
 ON pc.CustomerID = ur.CustomerID;
```

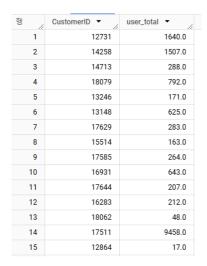
[결과 이미지를 넣어주세요]

Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(UnitPrice * Quantity)) AS user_total
FROM modulabs_project.data
WHERE Quantity > 0 AND UnitPrice > 0
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 `midyear-module-447402-s8.modulabs_project.user_rfm` AS
 rf.CustomerID AS CustomerID,
 rf.purchase_cnt,
 rf.item_cnt,
 rf.recency,
 ut.user_total,
 ROUND(ut.user_total / rf.purchase_cnt, 1) AS user_average
{\tt FROM\ `midyear-module-447402-s8.modulabs\_project.user\_rf`\ rf}
LEFT JOIN (
 -- 고객 별 총 지출액
 SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
 FROM `midyear-module-447402-s8.modulabs_project.data`
 WHERE Quantity > 0 AND UnitPrice > 0
 GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

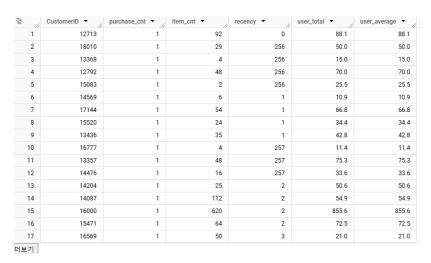
[결과 이미지를 넣어주세요]

RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM `midyear-module-447402-s8.modulabs_project.user_rfm`
```

[결과 이미지를 넣어주세요]



5-8. 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

[결과 이미지를 넣어주세요]

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 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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
   SELECT
     CustomerID,
     DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
     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]

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 `midyear-module-447402-s8.modulabs_project.user_data` AS

WITH TransactionInfo AS (
    SELECT
        CustomerID,
        COUNT(*) AS total_transactions,
        SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) AS cancel_frequency
    FROM `midyear-module-447402-s8.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
    GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(SAFE_DIVIDE(t.cancel_frequency, t.total_transactions), 2) AS
FROM `midyear-module-447402-s8.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
 u.CustomerID,
 u.recency,
 u.purchase_cnt AS frequency,
 u.user_total AS monetary,
 u.user_average,
 u.unique_products,
 u.average_interval,
 u.total_transactions,
 u.cancel_frequency,
 u.cancel_rate,
 ROUND(AVG(d.UnitPrice), 2) AS avg_unit_price,
 MAX(d.UnitPrice) AS max_unit_price,
 MIN(d.UnitPrice) AS min_unit_price,
 COUNT(DISTINCT d.StockCode) AS distinct_products,
 COUNT(DISTINCT EXTRACT(DAYOFWEEK FROM d.InvoiceDate)) AS active_days,
 MAX(d.InvoiceDate) AS last_purchase_date,
 MIN(d.InvoiceDate) AS first_purchase_date,
 {\tt DATE\_DIFF(MAX(d.InvoiceDate),\ MIN(d.InvoiceDate),\ DAY)\ AS\ customer\_lifetime\_days}
FROM `midyear-module-447402-s8.modulabs_project.user_data` AS u
LEFT JOIN `midyear-module-447402-s8.modulabs_project.data` AS d
ON u.CustomerID = d.CustomerID
GROUP BY
 \hbox{\tt u.CustomerID, u.recency, u.purchase\_cnt, u.user\_total, u.user\_average,}\\
 u.unique_products, u.average_interval, u.total_transactions,
 u.cancel_frequency, u.cancel_rate
ORDER BY u.user_total DESC
```

[결과 이미지를 넣어주세요]

```
| Number | N
```