PDA

고객을 세그먼테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
select *
from `mainquest.mainquestproject`
limit 10
;
```

InvoiceNo	StockCode	Description	Quantity
536414	22139	*null*	56
536545	21134	*null*	1
536546	22145	*null*	1
536547	37509	*null*	1
536549	85226A	*null*	1
536550	85044	*null*	1
536552	20950	*null*	1
536553	37461	*null*	3
536554	84670	*null*	23
536589	21777	*null*	-10

[결과 이미지를 넣어주세요]

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
select count(*)
from `mainquest.mainquestproject`
;
```

f0_
541909

[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 `mainquest.mainquestproject`
```

count_invoiceNO	count_stockcode	count_description	count_quantity	count_invoicedate	count_Unitprice	count_customerID	count_country
541909	541909	540455	541909	541909	541909	406829	541909

[결과 이미지를 넣어주세요]

5-4. 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_percent
FROM mainquest.mainquestproject
union all
SELECT
    'StockCode' AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `mainquest.mainquestproject`
union all
SELECT
    'description' AS column_name,
    ROUND(SUM(CASE WHEN description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM `mainquest.mainquestproject`
union all
SELECT
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percenta
FROM `mainquest.mainquestproject`
union all
SELECT
    'InvoiceDate' AS column_name,
    ROUND(SUM(CASE WHEN invoicedate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM `mainquest.mainquestproject`
union all
SELECT
    'UnitPrice' AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `mainquest.mainquestproject`
union all
SELECT
    'CustomerID' AS column_name,
    ROUND(SUM(CASE WHEN customerId IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percen
FROM `mainquest.mainquestproject`
union all
SELECT
    'country' AS column_name,
    ROUND(SUM(CASE WHEN country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentag
FROM `mainquest.mainquestproject`;
```

	column_name	missing_percentage
1	country	0.0
2	CustomerID	24.93
3	InvoiceNo	0.0
4	Quantity	0.0
5	StockCode	0.0
6	UnitPrice	0.0
7	InvoiceDate	0.0
8	description	0.27

[결과 이미지를 넣어주세요]

결측치 처리 전략

큰 비율의 누락된 값을 다른 값으로 대체하는 것은 분석에 상당한 편향을 주고 노이즈가 될 수 있습니다.

[행삭제 필요]

참고: 데이터 편향이란?
데이터가 특정 경향을 가지고 있어 전체적인 분석이나 결과가 왜곡되는 현상을 말합니다.

참고: 데이터 노이즈란?
데이터에 관련 없는 정보가 포함되어 있어 데이터의 질이 저하되고 분석의 정확도가 떨어지는 것을 말합니다.

- **`StockCode = '85123A'` 의 `Description` 을 추출하는 쿼리문을 작성하기**

```
select description
from `mainquest.mainquestproject`
where stockcode = '85123A'
group by description;
```

	description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	WHITE HANGING HEART T-LIG...

[결과 이미지를 넣어주세요]

결측치 처리

- **DELETE 구문을 사용하며, WHERE 절을 통해 데이터를 제거할 조건을 제시**

```
DELETE FROM main.quest
WHERE customerId is null or description is null
```

이 문으로 quest의 행 135,080개가 삭제되었습니다.

[결과 이미지를 넣어주세요]

5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- **중복된 행의 수를 세어보기**
  - **8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기**

```
with Coount as
(SELECT invoiceNO, stockcode, description, quantity, invoicedate,unitprice, customerid, country, COU
FROM main.quest
GROUP BY invoiceNO, stockcode, description, quantity, invoicedate,unitprice, customerid ,country
HAVING COUNT(*) > 1
ORDER BY count DESC)

select count(count) as duplicate from Coount;
```

	duplicate
1	4837

[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 hwang-447405.main.quest2

as select distinct * from hwang-447405.main.quest;

select count(*) from main.quest2;
```

f0_
401604

[결과 이미지를 넣어주세요]

5-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

- **고유(unique)한 `InvoiceNo` 의 개수를 출력하기**

```
select count(distinct invoiceNO)
from main.quest2;
```

	f0_
1	22190

[결과 이미지를 넣어주세요]

- **고유한 `InvoiceNo` 를 앞에서부터 100개를 출력하기**

```
select distinct invoiceNO
from main.quest2
limit 100;
```

	invoiceNO
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032

[결과 이미지를 넣어주세요]

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

```
select *
from main.quest2
where invoiceNO like 'C%'
limit 100;
```

	InvoiceNo	StockCode	Descripti
1	C575531	22960	JAM MA
2	C558080	22840	ROUND C
3	C558080	22847	BREAD B
4	C554983	47590B	PINK HA
5	C554983	47590A	BLUE HA

[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(invoiceno) * 100 , 1)
FROM main.quest2;
```

	f0_
1	2.2

[결과 이미지를 넣어주세요]

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
select count(distinct stockcode)
from main.quest2;
```

	f0_
1	3684

[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 main.quest2
group by stockcode
ORDER BY sell_cnt DESC
limit 10
;
```

	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346

[결과 이미지를 넣어주세요]

- **`StockCode`의 문자열 내 숫자의 길이를 구해보기**

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM main.quest2
)
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;
```

	number_count	stock_cnt
1	5	3676
2	0	7
3	1	1

[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 main.quest2
)
WHERE number_count = 0 or number_count = 1
;
```

	StockCode	number_count
1	POST	0
2	M	0
3	PADS	0
4	D	0
5	BANK CHARGES	0
6	DOT	0

[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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
select round(sum(case when LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1
from main.quest2;
```

	f0_
1	0.48

[결과 이미지를 넣어주세요]

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM main.quest3
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM main.quest3 )
WHERE number_count = 0 or number_count = 1);
```

이 문으로 quest4의 행 1,915개가 삭제되었습니다.

[결과 이미지를 넣어주세요]

`Description` 살펴보기

- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS description_cnt
FROM `main.quest3`
group by description
order by description_cnt desc
limit 30;
```

	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345

[결과 이미지를 넣어주세요]

- **대소문자가 혼합된 Description이 있는지 확인하기**

```
SELECT DISTINCT Description
FROM `main.quest3`
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

	Description
1	BAG 125g SWIRLY MARBLES
2	3 TRADITIONAl BISCUIT CUTT…
3	BAG 250g SWIRLY MARBLES
4	ESSENTIAL BALM 3.5g TIN IN …
5	FOLK ART GREETING CARD,pa…
6	BAG 500g SWIRLY MARBLES

[결과 이미지를 넣어주세요]

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM main.quest3
WHERE description = 'Next Day Carriage'
or description = 'High Resolution Image';
```

이 문으로 quest3의 행 83개가 삭제되었습니다.

[결과 이미지를 넣어주세요]

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE main.quest5 AS
SELECT
  * EXCEPT (Description),
  upper(description) AS Description
FROM main.quest3;


SELECT DISTINCT Description
FROM `main.ques5`
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

표시할 데이터가 없습니다.

[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 main.quest5;
```

min_price	max_price	avg_price
0.0	649.5	2.904956757405...

[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_quantity
FROM main.quest5
WHERE unitprice = 0
```

	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

[결과 이미지를 넣어주세요]

- **`UnitPrice = 0`를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE main.quest6 AS
SELECT *
FROM main.quest5
WHERE unitprice > 0


[결과이미지]
SELECT min(unitprice) AS min_price,
max(unitprice) AS max_price,
avg(unitprice) AS avg_price
FROM main.quest6;
```

	min_price	max_price	avg_price
1	0.03	649.5	2.905196672447...

[결과 이미지를 넣어주세요]

5-7. RFM 스코어

Recency

- **`InvoiceDate` 컬럼을 연월일 자료형으로 변경하기**

```
SELECT date(invoicedate) AS InvoiceDay, *
FROM main.quest6;
```

	InvoiceDay	InvoiceNo	StockCode
1	2011-11-03	574301	22750
2	2011-11-03	574301	23514
3	2011-11-03	574301	22086
4	2011-11-03	574301	23240
5	2011-11-03	574301	84879

[결과 이미지를 넣어주세요]

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
   # [[YOUR QUERY]] AS most_recent_date,
   # [[YOUR QUERY]] AS InvoiceDay,
   *
FROM project_name.modulabs_project.data;


select
max(date(invoicedate)) over () as most_recent_date,
date(invoicedate) as invoiceday
from main.quest6;

>> over 는 최종 같은 값은 이어 붙여주는 용도라고 생각을 하면 될듯..?
[찐 어려움]
```

	most_recent_date	invoiceday
1	2011-12-09	2011-11-03
2	2011-12-09	2011-11-07
3	2011-12-09	2011-10-14
4	2011-12-09	2011-11-25
5	2011-12-09	2011-07-01
6	2011-12-09	2011-07-01

[결과 이미지를 넣어주세요]

- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM main.quest6
  GROUP BY CustomerID
```

	CustomerID	InvoiceDay
1	12544	2011-11-10
2	13568	2011-06-19
3	13824	2011-11-07
4	14080	2011-11-07
5	14336	2011-11-23
6	14592	2011-11-04

[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 main.quest6
  GROUP BY CustomerID
);
```

	CustomerID	recency
1	17171	289
2	15657	22
3	14894	85
4	13103	39
5	13359	59
6	12599	29

[결과 이미지를 넣어주세요]

- **최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 `user_r` 이라는 이름의 테이블로 저장하기**

```
CREATE OR REPLACE TABLE main.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM main.quest6
```

```
  GROUP BY CustomerID
);
```

	CustomerID	recency
1	17364	0
2	12518	0
3	12713	0
4	15344	0
5	13069	0
6	17315	0
7	14446	0
8	12985	0
9	17581	0
10	16954	0

[결과 이미지를 넣어주세요]

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  customerID,
  count(invoiceno)
FROM main.quest6
group by customerID
```

	customerID	f0_
1	12544	19
2	13568	43
3	13824	46
4	14080	4
5	14336	90
6	14592	158

[결과 이미지를 넣어주세요]

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  customerID,
```

```
 sum(quantity) as item_cnt
FROM main.quest6
group by customerID
```

	customerID	f0_
1	12544	130
2	13568	66
3	13824	768
4	14080	48
5	14336	1759
6	14592	407

[결과 이미지를 넣어주세요]

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

```
CREATE OR REPLACE TABLE main.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
  customerID,
  count(invoiceno) AS purchase_cnt
FROM main.quest6
group by customerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
  customerID,
 sum(quantity) AS item_cnt
FROM main.quest6
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 main.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

	CustomerID	purchase_cnt	item_cnt	recency
1	17364	409	2671	0
2	17490	85	1022	0
3	17581	451	5849	0
4	14397	95	1852	0
5	15804	273	2513	0
6	15910	266	1013	0
7	12985	78	1413	0
8	16626	184	2670	0
9	17315	482	3805	0
10	16705	284	5458	0

[결과 이미지를 넣어주세요]

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  round(sum(unitprice),1) AS user_total
FROM main.quest6
group by customerID;
```

	CustomerID	user_total
1	12544	54.3
2	13568	130.6
3	13824	126.6
4	14080	3.8
5	14336	145.4
6	14592	323.6

[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 project_name.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
```

```
  ut.user_total,
  round(ut.user_total / rf.purchase_cnt) AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
  round(sum(quantity*uniprice),1)
  from main.quset6
  group by customerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

item_cnt	recency	user_total	user_average
856	0	1005.6	4.0
1013	0	1228.9	5.0
23516	0	29820.0	7.0
5454	0	3713.1	8.0
2671	0	4437.2	11.0

[결과 이미지를 넣어주세요]

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select *
from main.user_rfm;
```

	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	14446	276	856	0	1005.6	4.0
2	15910	266	1013	0	1228.9	5.0
3	12748	4440	23516	0	29820.0	7.0
4	13069	469	5454	0	3713.1	8.0

[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**

```
CREATE OR REPLACE TABLE main.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM main.quest6
  GROUP BY CustomerID
```

```
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM main.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	13747	1	8	373	79.6	80.0	1
2	18273	3	80	2	204.0	68.0	1
3	16998	5	74	149	287.5	58.0	1
4	13185	1	12	267	71.4	71.0	1
5	16737	1	288	53	417.6	418.0	1
6	17925	1	72	372	244.1	244.0	1
7	15389	1	400	172	500.0	500.0	1
8	14679	1	-1	371	-2.5	-3.0	1
9	18268	2	0	134	0.0	0.0	1
10	15562	1	39	351	134.6	135.0	1

[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 main.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)
    FROM
      main.quest6
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM main.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

3. 구매 취소 경향성 [못했음 ...어려움]

- **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 main.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(CASE WHEN InvoiceNO = 'C%' THEN 1 END) AS cancel_frequency,
    COUNT(CASE WHEN invoiceNO = 'C%' THEN 1 END) / COUNT(*) AS cancel_rate
  FROM main.quest6
  GROUP BY CustomerID
)

SELECT
  u.CustomerID,
  u.*,
  t.cancel_frequency,
  ROUND(t.cancel_rate, 2) AS cancel_rate
FROM main.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 main.user_data
```

[결과 이미지를 넣어주세요]

좋았던 점

- 다양한 문제들을 풀 수 있는 기회이기 때문에 좋았음
- 문제를 스스로 풀 수 있는 기회여서 좋았긴 하지만 챗GPT 활용 능력도 같이 향상 되었음(좋은점 맞음)

아쉬웠던 점

- 현실에 마주보게 되면서 SQL에 자신이 없어짐
- 다양한 방법으로 풀려고 노력했지만 이해보다는 풀이에 집중 했던 것 같음