고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

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

  [결과 이미지를 넣어주세요]

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

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

  [결과 이미지를 넣어주세요]

데이터 수 세기

- **COUNT 함수를 사용해서, 각 컬럼별 데이터 포인트의 수를 세어 보기**

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

  [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**

```
⭐ 전체 행 수를 기준으로 각 컬럼별 결측치 비율 계산 (백분율)
SELECT 'InvoiceNo' AS column_name,
    ROUND(COUNT(CASE WHEN InvoiceNo IS NULL THEN 1 END) / C
OUNT(*) * 100, 2) AS null_ratio_percent
FROM `automatic-rock-473602-a0.modulabs_project.data`

UNION ALL
SELECT 'StockCode',
    ROUND(COUNT(CASE WHEN StockCode IS NULL THEN 1 END) / C
OUNT(*) * 100, 2)
FROM `automatic-rock-473602-a0.modulabs_project.data`

UNION ALL
SELECT 'Description',
    ROUND(COUNT(CASE WHEN Description IS NULL THEN 1 END) / C
OUNT(*) * 100, 2)
FROM `automatic-rock-473602-a0.modulabs_project.data`

UNION ALL
SELECT 'Quantity',
    ROUND(COUNT(CASE WHEN Quantity IS NULL THEN 1 END) / CO
UNT(*) * 100, 2)
FROM `automatic-rock-473602-a0.modulabs_project.data`

UNION ALL
SELECT 'InvoiceDate',
    ROUND(COUNT(CASE WHEN InvoiceDate IS NULL THEN 1 END) /
COUNT(*) * 100, 2)
FROM `automatic-rock-473602-a0.modulabs_project.data`

UNION ALL
SELECT 'UnitPrice',
```

```
    ROUND(COUNT(CASE WHEN UnitPrice IS NULL THEN 1 END) / CO
UNT(*) * 100, 2)
FROM `automatic-rock-473602-a0.modulabs_project.data`

UNION ALL
SELECT 'CustomerID',
    ROUND(COUNT(CASE WHEN CustomerID IS NULL THEN 1 END) /
COUNT(*) * 100, 2)
FROM `automatic-rock-473602-a0.modulabs_project.data`

UNION ALL
SELECT 'Country',
    ROUND(COUNT(CASE WHEN Country IS NULL THEN 1 END) / COU
NT(*) * 100, 2)
FROM `automatic-rock-473602-a0.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	column_name	null_ratio_percent
1	InvoiceDate	0.0
2	Country	0.0
3	Description	0.27
4	UnitPrice	0.0
5	Quantity	0.0
6	CustomerID	24.93
7	InvoiceNo	0.0
8	StockCode	0.0

결측치 처리 전략

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

```
SELECT DISTINCT(Description)
FROM `automatic-rock-473602-a0.modulabs_project.data`
```

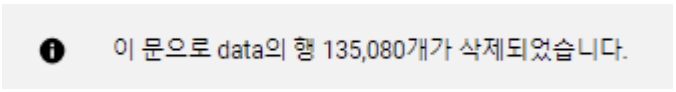
```
WHERE StockCode = '85123A'
ORDER BY Description ASC;
```

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

결측치 처리

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

```
DELETE FROM `automatic-rock-473602-a0.modulabs_project.data`
WHERE Description IS NULL
OR CustomerID IS NULL;
```



11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
✴️ 8개의 컬럼에 그룹함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어 중복된 행의 수 확인
SELECT COUNT(*) AS DuplicateRowsCount
```

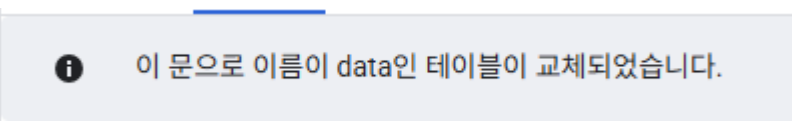
```
FROM (SELECT *
        , COUNT(*) AS row_count
    FROM `automatic-rock-473602-a0.modulabs_project.data`
    GROUP BY InvoiceNo
        , StockCode
        , Description
        , Quantity
        , InvoiceDate
        , UnitPrice
        , CustomerID
        , Country
    HAVING row_count > 1
);
```

행	DuplicateRowsCo...
1	4837

중복값 처리

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

```
✴️ CREATE OR REPLACE TABLE 구문을 활용하여 모든 컬럼(*)을 DISTINCT 한 데이터로 업데이트
CREATE OR REPLACE TABLE
  `automatic-rock-473602-a0.modulabs_project.data` AS
SELECT
  DISTINCT *
FROM
  `automatic-rock-473602-a0.modulabs_project.data`
```



11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```
✴️ 고유한(unique) InvoiceNO의 개수 출력
SELECT COUNT(DISTINCT InvoiceNO)
FROM `automatic-rock-473602-a0.modulabs_project.data`;
```

행	f0_
1	22190

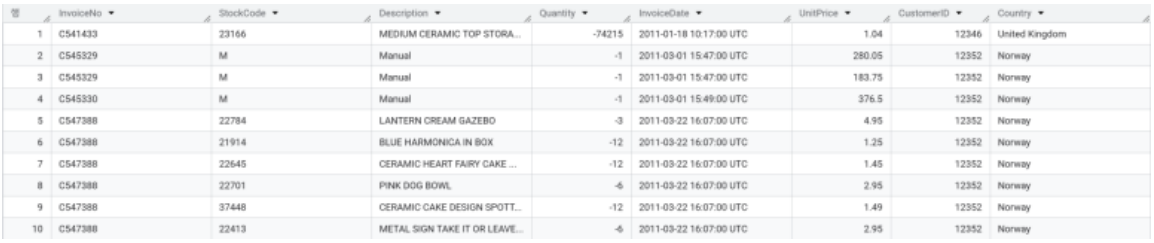
- **고유한 `InvoiceNo`를 앞에서부터 100개를 출력하기**

```
✴️ 고유한(unique) InvoiceNO 100개 출력
SELECT DISTINCT InvoiceNO
FROM `automatic-rock-473602-a0.modulabs_project.data`
LIMIT 100
```

행	InvoiceNO
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318

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

```
✴️ InvoiceNO가 'C'로 시작하는 행을 필터링하여 100개 출력
SELECT *
FROM `automatic-rock-473602-a0.modulabs_project.data`
WHERE InvoiceNO LIKE 'C%'
LIMIT 100;
```

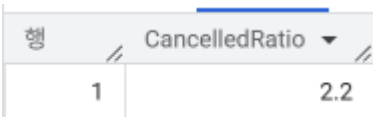


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
6	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
7	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
8	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
9	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
10	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway

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

```
✴️ 구매 건 상태가 Canceled인 데이터의 비율(%) (소수점 첫 번째 자리까지)
SELECT
    ROUND(
        (SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) * 10
0.0) / COUNT(*),
```

```
        1
    ) AS CancelledRatio
FROM
    `automatic-rock-473602-a0.modulabs_project.data`;
```

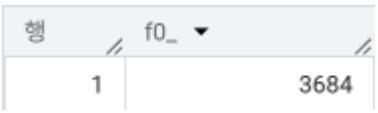


행	CancelledRatio
1	2.2

StockCode 살펴보기

- **고유한 StockCode의 개수를 출력하기**

```
✳️ 고유한(unique) StockCode의 개수 출력
SELECT COUNT(DISTINCT StockCode)
FROM `automatic-rock-473602-a0.modulabs_project.data`;
```



행	f0_
1	3684

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

```
✳️ StockCode별 빈도 상위 10개 출력
SELECT StockCode
    , COUNT(*) AS sell_cnt
FROM `automatic-rock-473602-a0.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

행	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
9	22197	1110
10	23203	1108

- **`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

```
⭐ StockCode의 문자열 내 숫자 길이 확인
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM `automatic-rock-473602-a0.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
```

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 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
✴️ 전체 데이터 수 대비 해당 코드 값들을 가지고 있는 데이터(%) (소수점 둘 째 자리까지)
SELECT
  ROUND(
    (SUM(
      CASE
        -- StockCode에 숫자가 0개 또는 1개인 경우 (REGEXP_REPLACE를 사용한 로직 재사용)
        WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1) THEN 1
        ELSE 0
      END
    ) * 100.0) / COUNT(*),
    2
  ) AS StockCodeRatio
FROM
  `automatic-rock-473602-a0.modulabs_project.data`
```

행	StockCodeRatio
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
✴️ 숫자가 0-1개인 값들에는 어떤 코드 데이터셋에서 제외
DELETE FROM `automatic-rock-473602-a0.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 `automatic-rock-473602-a0.modulabs_project.data`
)
WHERE number_count IN (0, 1)
);
```

Description 살펴보기

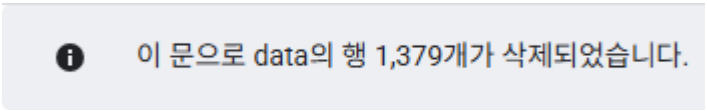
- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
✳️ 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력
SELECT Description
    , COUNT(*) AS description_cnt
FROM `automatic-rock-473602-a0.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

행	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
7	SET OF 3 CAKE TINS PANTRY D...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE ...	1062
10	SPOTTY BUNTING	1026

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
✳️ 서비스 관련 정보를 포함하는 행들을 제거
DELETE FROM `automatic-rock-473602-a0.modulabs_project.data`
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
✳️ 대소문자를 혼합하고 있는 데이터를 대문자로 표준화
UPDATE `automatic-rock-473602-a0.modulabs_project.data`
SET Description = UPPER(Description)
WHERE 1=1
```

UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
✳️ UnitPrice의 최솟값, 최댓값, 평균
SELECT MIN(UnitPrice) AS min_unitprice
, MAX(UnitPrice) AS max_unitprice
, AVG(UnitPrice) AS avg_unitprice
FROM automatic-rock-473602-a0.modulabs_project.data;
```

행	min_unitprice	max_unitprice	avg_unitprice
1	0.0	649.5	2.910256885340...

- **단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기**

```
✳️ 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균값
SELECT COUNT(Quantity) AS cnt_quantity
, MIN(Quantity) AS min_quantity
, MAX(QUantity) AS max_quantity
, AVG(Quantity) AS avg_quantity
FROM automatic-rock-473602-a0.modulabs_project.data
WHERE UnitPrice = 0;
```

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

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

```
✴️ UnitPrice = 0인 데이터 제거
CREATE OR REPLACE TABLE automatic-rock-473602-a0.modulabs_project.data AS
SELECT *
FROM automatic-rock-473602-a0.modulabs_project.data
WHERE UnitPrice > 0;
```

11-7. RFM 스코어

Recency

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

```
✴️ InvoviceDate컬럼을 'YYYY-MM-DD HH:MM:SS'형태에서 'YYYY-MM-DD'형태로 변환
CREATE OR REPLACE TABLE
automatic-rock-473602-a0.modulabs_project.data AS
SELECT
* REPLACE(DATE(InvoiceDate) AS InvoiceDate)
FROM
automatic-rock-473602-a0.modulabs_project.data
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536370	10002	INFLATABLE POLITICAL GLOBE	48	2010-12-01	0.85	12583	France
2	536382	10002	INFLATABLE POLITICAL GLOBE	12	2010-12-01	0.85	16098	United Kingdom
3	536863	10002	INFLATABLE POLITICAL GLOBE	1	2010-12-03	0.85	17967	United Kingdom
4	537047	10002	INFLATABLE POLITICAL GLOBE	1	2010-12-05	0.85	13069	United Kingdom
5	537227	10002	INFLATABLE POLITICAL GLOBE	24	2010-12-06	0.85	17677	United Kingdom
6	537770	10002	INFLATABLE POLITICAL GLOBE	12	2010-12-08	0.85	15529	United Kingdom
7	538069	10002	INFLATABLE POLITICAL GLOBE	8	2010-12-09	0.85	16795	United Kingdom
8	538086	10002	INFLATABLE POLITICAL GLOBE	10	2010-12-09	0.85	12872	United Kingdom
9	538093	10002	INFLATABLE POLITICAL GLOBE	12	2010-12-09	0.85	12682	France
10	538167	10002	INFLATABLE POLITICAL GLOBE	12	2010-12-09	0.85	14713	United Kingdom

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
📏 가장 최근 구매 일자 확인
SELECT
  MAX(InvoiceDate) AS most_recent_date
FROM `automatic-rock-473602-a0.modulabs_project.data`;
```

행	most_recent_date
1	2011-12-09

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

```
⭐ 유저별로 가장 큰 InvoiceDate를 찾아서 가장 최근 구매일로 저장
SELECT
    CustomerID
  , MAX(InvoiceDate) AS InvoiceDay
FROM `automatic-rock-473602-a0.modulabs_project.data`
GROUP BY CustomerID
ORDER BY CustomerID ASC;
```

행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17

- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

```
✳️ 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이 계산
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `automatic-rock-473602-a0.modulabs_project.data`
  GROUP BY CustomerID
);
```

행	CustomerID	recency
1	17994	121
2	16503	106
3	15494	21
4	13521	1
5	16161	1
6	13060	253
7	15533	19
8	12390	79
9	12527	81
10	16306	213

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

```
✳️ 최종 데티어셋에 필요한 데이터들을 각각 정제해서 이어붙이고,
지금까지의 결과를 user_r이라는 테이블로 저장
```

```
CREATE OR REPLACE TABLE
  `automatic-rock-473602-a0.modulabs_project.user_r` AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS rec
ency
FROM
  (
    SELECT
      CustomerID,
      MAX(InvoiceDate) AS InvoiceDay
    FROM
      `automatic-rock-473602-a0.modulabs_project.data`
    GROUP BY
      CustomerID
  );
```

행	CustomerID	recency
1	17001	0
2	12748	0
3	17581	0
4	15910	0
5	16446	0
6	15344	0
7	13426	0
8	17315	0
9	17389	0
10	16626	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
✳️ 1. 전체 거래 건수 계산
SELECT
  CustomerID
, COUNT(DISTINCT InvoiceNO) AS purchase_cnt
FROM `automatic-rock-473602-a0.modulabs_project.data`
GROUP BY CustomerID;
```

행	CustomerID	purchase_cnt
1	12583	17
2	16098	7
3	17967	1
4	13069	27
5	17677	43
6	15529	12
7	16795	1
8	12872	2
9	12682	31
10	14713	12

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
✳️ 2. 구매한 아이템의 총 수량 계산
SELECT
  CustomerID
, SUM(Quantity) AS item_cnt
FROM `automatic-rock-473602-a0.modulabs_project.data`
GROUP BY CustomerID;
```

행	CustomerID	item_cnt
1	12583	4978
2	16098	613
3	17967	73
4	13069	5430
5	17677	9710
6	15529	3417
7	16795	239
8	12872	319
9	12682	5485
10	14713	1572

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

```
✴️ 1과 2의 결과를 합쳐서 user_rf라는 이름의 테이블에 저장
CREATE OR REPLACE TABLE `automatic-rock-473602-a0.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
      CustomerID
    , COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `automatic-rock-473602-a0.modulabs_project.data`
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
      CustomerID
    , SUM(Quantity) AS item_cnt
  FROM `automatic-rock-473602-a0.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 `automatic-rock-473602-a0.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	15520	1	314	1
3	13298	1	96	1
4	13436	1	76	1
5	14569	1	79	1
6	15195	1	1404	2
7	14204	1	72	2
8	15471	1	256	2
9	15992	1	17	3
10	12650	1	250	3

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
✴️ 1. 고객별 총 지출액 계산 - 소수점 첫 째자리에서 반올림
SELECT
```

```
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS total_spent
FROM
    `automatic-rock-473602-a0.modulabs_project.data`
GROUP BY
    CustomerID;
```

행	CustomerID	total_spent
1	12583	6629.0
2	16098	2006.0
3	17967	112.0
4	13069	3704.0
5	17677	16176.0
6	15529	3911.0
7	16795	415.0
8	12872	600.0
9	12682	11020.0
10	14713	2596.0

- **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**

```
✴️ 2. 고객별 평균 거래 금액 계산
1) data 테이블을 user_rf 테이블과 조인(LEFT JOIN) 한 후
2) purchase_cnt로 나누어서 3) user_rfm 테이블로 저장

CREATE OR REPLACE TABLE automatic-rock-473602-a0.modulabs_pro
ject.user_rfm AS
SELECT
rf.CustomerID AS CustomerID,
rf.purchase_cnt,
rf.item_cnt,
```

```
rf.recency,
ut.user_total,
-- 총 지출액(user_total)을 거래 건수(purchase_cnt)로 나누어 평균 거래 금액 계산
ROUND(ut.user_total / rf.purchase_cnt) AS user_average
FROM automatic-rock-473602-a0.modulabs_project.user_rf AS rf
LEFT JOIN (
-- 고객 별 총 지출액 계산
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice)) AS user_total
FROM automatic-rock-473602-a0.modulabs_project.data
GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13436	1	76	1	197.0	197.0
3	13298	1	96	1	360.0	360.0
4	14569	1	79	1	227.0	227.0
5	15520	1	314	1	343.0	343.0
6	14204	1	72	2	151.0	151.0
7	15195	1	1404	2	3861.0	3861.0
8	15471	1	256	2	454.0	454.0
9	17914	1	457	3	329.0	329.0
10	12478	1	233	3	546.0	546.0

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
✴️ 최종 user_rfm 테이블을 출력
SELECT
  *
```

```
FROM
  `automatic-rock-473602-a0.modulabs_project.user_rfm`;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13436	1	76	1	197.0	197.0
3	13298	1	96	1	360.0	360.0
4	14569	1	79	1	227.0	227.0
5	15520	1	314	1	343.0	343.0
6	14204	1	72	2	151.0	151.0
7	15195	1	1404	2	3861.0	3861.0
8	15471	1	256	2	454.0	454.0
9	17914	1	457	3	329.0	329.0
10	12478	1	233	3	546.0	546.0

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**

```
✳️ 1) 고객별 구매한 상품의 고유한 수 계산
2) user_rfm 테이블과 결과를 합치고, 3) user_data라는 이름의 테이블에 저장

CREATE OR REPLACE TABLE `automatic-rock-473602-a0.modulabs_projec
t.user_data` AS
WITH unique_products AS (
  -- (1) 고객별 고유 상품 수 계산
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `automatic-rock-473602-a0.modulabs_project.data`
  GROUP BY CustomerID
)
```

```
SELECT
  ur.*,
  up.* EXCEPT (CustomerID) -- CustomerID를 제외하고 unique_products 컬럼만 선택
FROM `automatic-rock-473602-a0.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
  ON ur.CustomerID = up.CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...
1	14424	1	48	17	322.0	322.0	1
2	14576	1	12	372	35.0	35.0	1
3	17102	1	2	261	26.0	26.0	1
4	13307	1	4	120	15.0	15.0	1
5	16257	1	1	176	22.0	22.0	1
6	15668	1	72	217	76.0	76.0	1
7	16995	1	-1	372	-1.0	-1.0	1
8	13841	1	100	252	85.0	85.0	1
9	13703	1	10	318	100.0	100.0	1
10	16093	1	20	106	17.0	17.0	1

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**

```
▌평균 구매 소요 일수를 계산하고, 그 결과를 user_data에 통합
CREATE OR REPLACE TABLE automatic-rock-473602-a0.modulabs_project.user_data AS
WITH purchase_intervals AS (
-- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
SELECT
CustomerID,
-- interval_의 평균을 계산하고 소수점 둘째 자리에서 반올림.
-- 단, 고객의 구매가 1회뿐이라 interval_이 NULL일 경우 0으로 처리.
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 (interval_) 계산
SELECT
CustomerID,
-- 현재 InvoiceDate에서 바로 직전 InvoiceDate를 빼서 일수 차이(DAY)를 계산
DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerI
D ORDER BY InvoiceDate), DAY) AS interval_
FROM
automatic-rock-473602-a0.modulabs_project.data
WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

-- 기존 user_data에 계산된 평균 소요 일수 통합
SELECT
u.*,
pi.* EXCEPT (CustomerID)
FROM automatic-rock-473602-a0.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...
1	15070	1	36	372	106.0	106.0	1
2	15389	1	400	172	500.0	500.0	1
3	16579	1	-12	365	-31.0	-31.0	1
4	17307	1	-144	365	-153.0	-153.0	1
5	16257	1	1	176	22.0	22.0	1
6	12943	1	-1	301	-4.0	-4.0	1
7	17331	1	16	123	175.0	175.0	1
8	13747	1	8	373	80.0	80.0	1
9	18068	1	6	289	102.0	102.0	1
10	18141	1	-12	360	-35.0	-35.0	1

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)

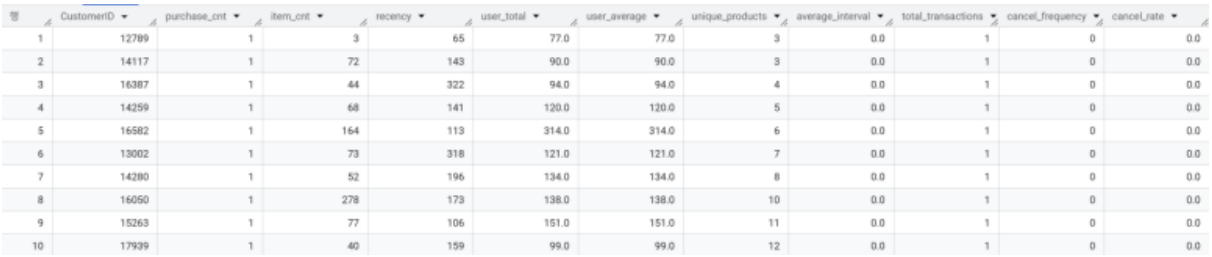
```
✳️ 취소 빈도와 취소 비율을 계산하고, 결과를 user_data에 통합 - 취소 비율은 소수점 둘 째 자리
CREATE OR REPLACE TABLE `automatic-rock-473602-a0.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions, -- (1) 총 거래 건수
    COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo ELSE NULL END) AS cancel_frequency -- (2) 취소 거래 건수
  FROM `automatic-rock-473602-a0.modulabs_project.data`
  GROUP BY CustomerID -- (3) 고객별로 그룹화
)

SELECT
    u.*,
    t.* EXCEPT(CustomerID),
    -- (4) 취소 비율 계산: 취소 빈도 / 총 거래 건수 (소수점 둘째 자리까지 반올림)
    ROUND(t.cancel_frequency * 1.0 / t.total_transactions, 2) AS cancel_rate
FROM `automatic-rock-473602-a0.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID; -- (5) CustomerID로 조인
```

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

```
 최종적으로 user_data를 출력
SELECT
   *
FROM
   `automatic-rock-473602-a0.modulabs_project.user_data`
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	12789	1	3	65	77.0	77.0	3	0.0	1	0	0.0
2	14117	1	72	143	90.0	90.0	3	0.0	1	0	0.0
3	16387	1	44	322	94.0	94.0	4	0.0	1	0	0.0
4	14259	1	68	141	120.0	120.0	5	0.0	1	0	0.0
5	16582	1	164	113	314.0	314.0	6	0.0	1	0	0.0
6	13002	1	73	318	121.0	121.0	7	0.0	1	0	0.0
7	14280	1	52	196	134.0	134.0	8	0.0	1	0	0.0
8	16050	1	278	173	138.0	138.0	10	0.0	1	0	0.0
9	15263	1	77	106	151.0	151.0	11	0.0	1	0	0.0
10	17939	1	40	159	99.0	99.0	12	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep : 내가 어느 정도까지 이해를 하고 있는지 확인 할 수 있었다.

Problem : 아직 덜 숙달되어서 시간도 정말 오래 걸렸고, AI의 도움을 많이 받았다.

Try : 이전 노션들에 있는 과제들을 풀면서 제대로 이해 및 숙달 후, 해당 프로젝트를 다시 진행 해봐야겠다.