이가현_고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
LIMIT 10;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART ...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS C...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WH...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTI...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION J...	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POL...	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	f0_
1	541909

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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	COUNT_UnitPrice	COUNT_Custome...	COUNT_Country
1	541909	541909	540455	541909	541909	541909	406829	541909

⇒ 누락된 컬럼 : Description, CustomerID

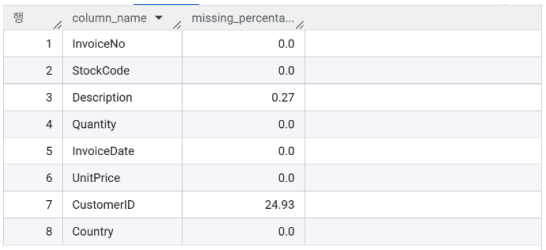
11-4. 데이터 전처리 방법(1): 결측치 제거

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
  'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
UNION ALL
SELECT
  'StockCode' AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
UNION ALL
SELECT
  'Description' AS column_name,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
UNION ALL
SELECT
  'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
UNION ALL
SELECT
  'InvoiceDate' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
UNION ALL
SELECT
  'UnitPrice' AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
UNION ALL
SELECT
  'CustomerID' AS column_name,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
UNION ALL
SELECT
  'Country' AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



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

⇒ Descrption : 0.27% 결측치

- 결측치 비율이 비교적 적음.
- 같은 제품 StockCode가 항상 같은 상세 설명 (Description) 을 가지고 있지 않음 → 데이터 일관성 문제

→ 결측치 : 같은 제품의 상세설명을 추출 및 채우기.

⇒ CustomerID : 24.93% 결측치. (약 4분의 1)

- CustomerID는 고객을 클러스터링 할 때, 필수적인 정보임.
- 누락 비율이 크기 때문에 다른 값으로 대체하는 것은 분석엔 편향과 노이즈가 될 수 있음.

- 고객 세그멘테이션이 목표이기 때문에, 고객 식별자 데이터는 정확해야 함.

→ 누락된 CustomerID의 해당 행을 제거.

```
SELECT column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(
   SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data UNION ALL
   SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data UNION ALL
   SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data UNION ALL
   SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data UNION ALL
   SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data UNION ALL
   SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data UNION ALL
   SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data UNION ALL
   SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM project_name.modulabs_project.data
) AS column_data;
```

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

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

⇒ 동일한 제품에 대한 상세 설명이 다름 → 데이터 일관성 문제.

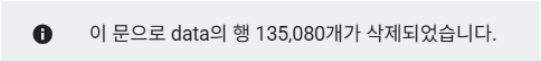
결측치 처리

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

```
DELETE
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
WHERE Description IS NULL
  OR CustomerID IS NULL;
```

[결과 이미지를 넣어주세요]

- 135,080개의 행이 삭제됨을 확인함.



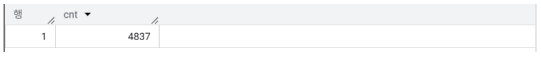
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT COUNt(*) AS cnt
FROM (
  SELECT *
  FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  GROUP BY 1,2,3,4,5,6,7,8
  HAVING COUNT(*) > 1
);
```

[결과 이미지를 넣어주세요]

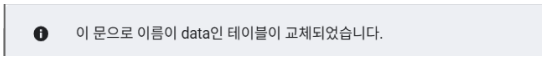


중복값 처리

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

```
CREATE OR REPLACE TABLE `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data` AS
SELECT DISTINCT *
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



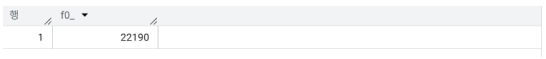
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]

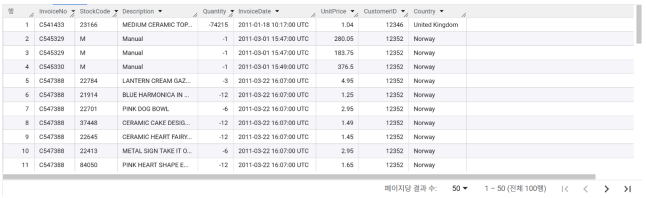
행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180

- ‘C’로 시작함 → 취소한 거래.

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

```
SELECT *
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

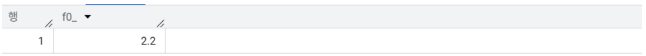


⇒ 취소된 거래 건들은 Quantity 가 음수이다.

- **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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



행	f0_
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode)
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	f0_
1	3684

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
GROUP BY 1
```

```
ORDER BY 2 DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

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

⇒ 대부분의 StockCode는 5~6자리 숫자와 문자 조합.

⇒ StockCode 중 'POST' 는 이상치.

- `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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
)
WHERE number_count IN (0, 1);
```

[결과 이미지를 넣어주세요]

행	StockCode	number_count
1	POST	0
2	M	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)

```
SELECT ROUND(
  COUNT(*) / (SELECT COUNT(*) FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`) * 100, 2
)
FROM (
  SELECT StockCode
  FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1)
);
```

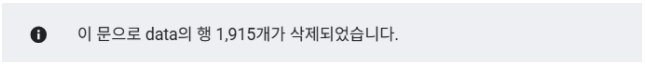
[결과 이미지를 넣어주세요]

행	f0_
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1)
);
```

[결과 이미지를 넣어주세요]



Description 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

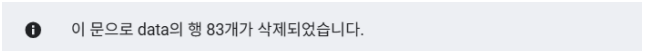
[결과 이미지를 넣어주세요]

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

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
WHERE UPPER(Description) LIKE '%NEXT DAY CARRIAGE%'
  OR UPPER(Description) LIKE '%HIGH RESOLUTION IMAGE%';
```

[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data` AS
SELECT * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

행	min_price	max_price	avg_price
1	0.0	649.5	2.9049567574059805

⇒ UnitPrice 가 0인 데이터 → 무료제품 or 오류 일 가능성.

- **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) A
S avg_quantity
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.51515151515139

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

```
CREATE OR REPLACE TABLE `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data` AS
SELECT *
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
WHERE UnitPrice > 0;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 테이블이 교체되었습니다.

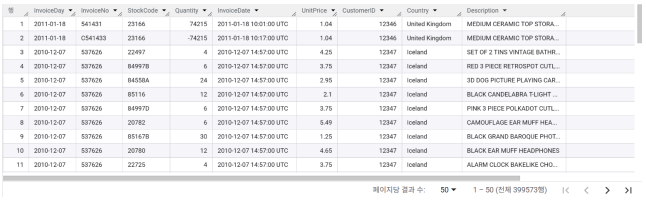
11-7. RFM 스코어

Recency

- **InvoiceDate 컬럼을 연월일 자료형으로 변경하기**

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

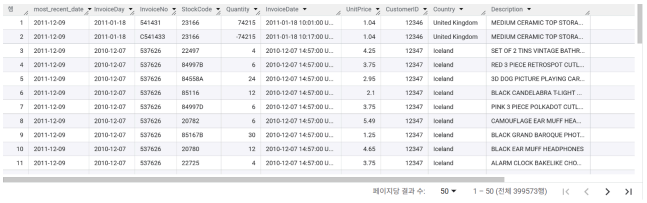
[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay, *
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`;
```

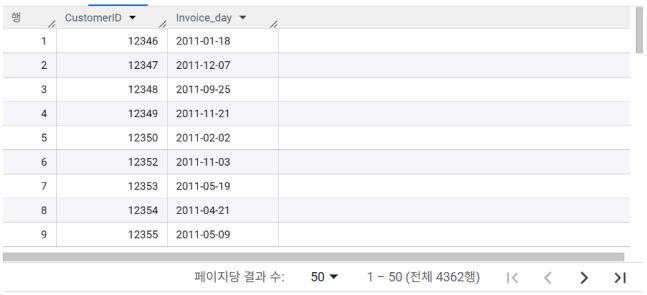
[결과 이미지를 넣어주세요]



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

```
SELECT CustomerID,
  MAX(DATE(InvoiceDate)) AS Invoice_day
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]



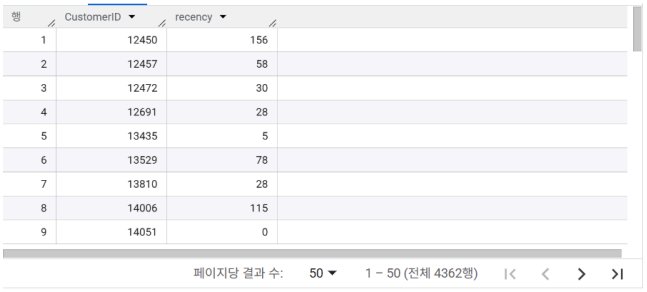
행	CustomerID	Invoice_day
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09

페이지당 결과 수: 50 1 – 50 (전체 4362행)

- **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-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]



행	CustomerID	recency
1	12450	156
2	12457	58
3	12472	30
4	12691	28
5	13435	5
6	13529	78
7	13810	28
8	14006	115
9	14051	0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

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

```
CREATE OR REPLACE TABLE `project-68da2f3e-5ce6-45c2-b5b.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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

행	CustomerID	recency
1	15804	0
2	13113	0
3	16954	0
4	12433	0
5	16705	0
6	17315	0
7	12680	0
8	13777	0
9	14441	0

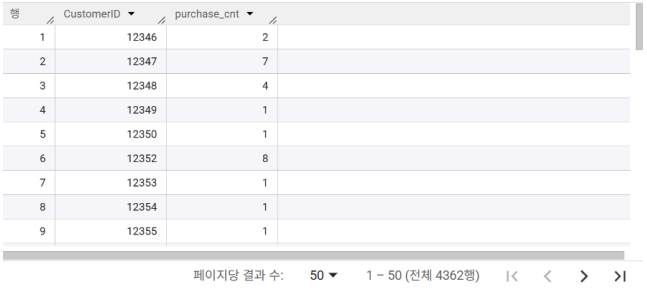
페이지당 결과 수: 50 1 – 50 (전체 4362행)

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]



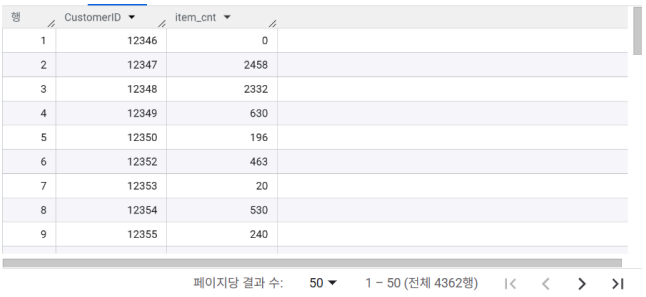
행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1
8	12354	1
9	12355	1

페이지당 결과 수: 50 1 – 50 (전체 4362행)

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT CustomerID,
  SUM(Quantity) AS item_cnt
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

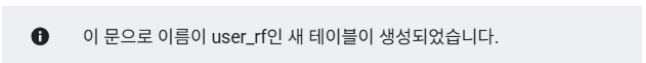


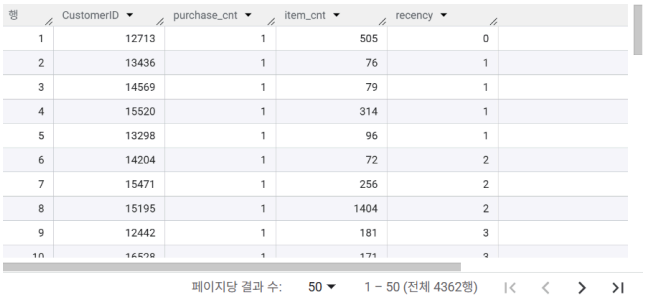
행	CustomerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240

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

```
CREATE OR REPLACE TABLE `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT CustomerID,
  SUM(Quantity) AS item_cnt
  FROM `project-68da2f3e-5ce6-45c2-b5b.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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]





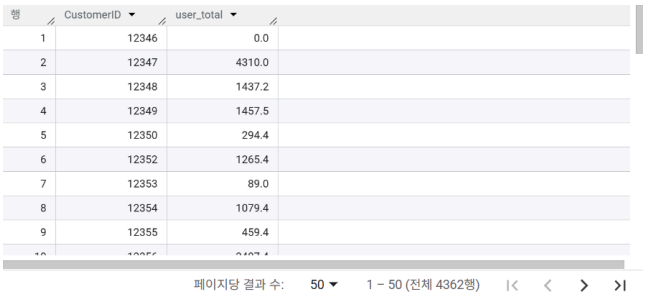
행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13436	1	76	1
3	14569	1	79	1
4	15520	1	314	1
5	13298	1	96	1
6	14204	1	72	2
7	15471	1	256	2
8	15195	1	1404	2
9	12442	1	181	3

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

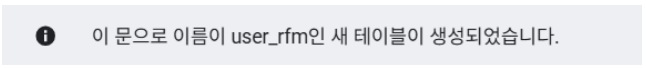


행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4

- **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-68da2f3e-5ce6-45c2-b5b.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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_rf` AS rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

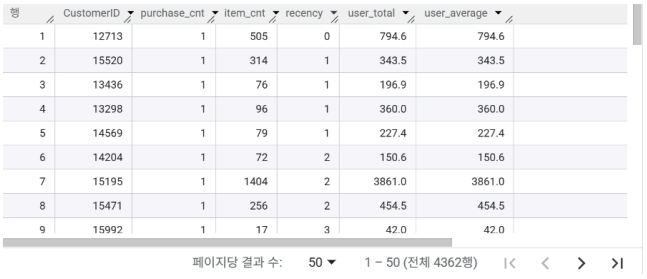


RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_rfm`;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.6	794.6
2	15520	1	314	1	343.5	343.5
3	13436	1	76	1	196.9	196.9
4	13298	1	96	1	360.0	360.0
5	14569	1	79	1	227.4	227.4
6	14204	1	72	2	150.6	150.6
7	15195	1	1404	2	3861.0	3861.0
8	15471	1	256	2	454.5	454.5
9	15992	1	17	3	42.0	42.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

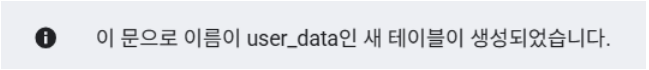
11-8. 추가 Feature 추출

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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_data` AS

WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
  ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]



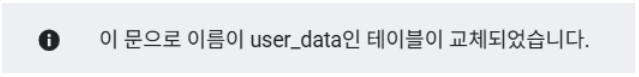
2. 평균 구매 주기

- **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 `project-68da2f3e-5ce6-45c2-b5b.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_
    FROM
      `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

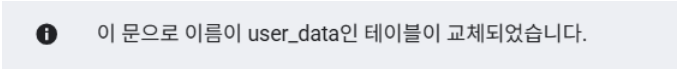


3. 구매 취소 경향성

- **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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT IF(InvoiceNo LIKE 'C%', InvoiceNo, NULL)) AS cancel_frequency
  FROM `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.data`
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  ROUND(t.cancel_frequency / t.total_transactions * 100, 2) AS cancel_rate
FROM `project-68da2f3e-5ce6-45c2-b5b.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 `project-68da2f3e-5ce6-45c2-b5b.modulabs_project.user_data`;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	18133	1	1350	212	931.5	931.5	1	0.0	1	0	0.0
2	18058	1	88	9	170.2	170.2	3	0.0	1	0	0.0
3	14589	1	5	371	39.8	39.8	3	0.0	1	0	0.0
4	14080	1	48	32	45.6	45.6	4	0.0	1	0	0.0
5	17148	1	82	51	124.9	124.9	9	0.0	1	0	0.0
6	13108	1	298	372	350.1	350.1	10	0.0	1	0	0.0
7	12551	1	100	357	168.0	168.0	10	0.0	1	0	0.0
8	15773	1	311	5	635.7	635.7	10	0.0	1	0	0.0
9	14806	1	330	75	193.4	193.4	11	0.0	1	0	0.0
10	17030	1	146	63	146.9	146.9	11	0.0	1	0	0.0
11	16684	1	74	377	217.3	217.3	14	0.0	1	0	0.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

최종 분석 및 인사이트

- 최종 user_data는 RFM + 구매 다양성 + 재방문 주기 + 취소 성향 까지 포함된 고객 테이블.

1. **단발성 고객이 매우 많다.**
   - purchase_cnt = 1 또는 average_interval = 0 인 고객이 많다.

     →1회 구매 이후 재구매가 없음.

     ⇒ 이탈 고객 분석 및 재구매 타킷으로 매우 중요함.
2. **구매 패턴이 극단적으로 나뉜다.**
   - purcahse_cnt = 1 인 고객 → item_cnt = 1350, user_total = 900
   - purcahse_cnt는 높은 고객 → user_average는 낮음.

   ⇒ 대량으로 한번 구매하는 유형 vs 소액으로 자주 반복 구매하는 유형
3. **취소 관련 변수는 대부분 0이다.**
   - cancel_rate = 0인 고객이 다수 있음.
   - 일부 고객만 취소 빈도/비율이 존재한다.

회고

[회고 내용을 작성해주세요]

- 이번 고객 세그먼테이션 프로젝트에서 Kaggle에 있는 이커머스 데이터를 활용하여 고객 세그먼테이션을 목표로 데이터 전처리부터 RFM 분석, 그리고 추가 feature 생성까지의 전 과정을 경험하였다.
  단순히 모델을 적용하는 것이 아니라, 분석 목적에 맞는 데이터셋을 직접 설계 및 정제하는 과정의 중요성을 느낄 수 있었다.

Keep :

- 데이터 분석을 바로 시작하지 않고, 결측치, 이상치, 중복값, 서비스성 데이터 등 데이터 품질을 먼저 점검하고 정제한 과정이 흥미로웠다.
- RFM 분석을 할 때, 단순히 RFM 계산만 하지 않고
  구매 수량, 평균 구매 주기, 상품 다양성, 취소 빈도 및 취소 비율 등의 추가 feature를 설계하여 고객 행동을 다양한 시각으로 해석하는 것을 배웠고, 이를 다양한 목표의 데이터 분석을 할 때 활용하도록 노력할 것이다.
- 모든 과정을 SQL로 단계별로 분리하여 TABLE을 생성함으로써, 분석 흐름이 명확하고 재현 가능한 파이프라인을 구성한 점이 매우 인상깊었다.

Problem :

- 데이터 결측치를 처리할 때, 처음에는 어떤 데이터를 제거해야 하는지에 대한 판단 기준이 명확하지 않아 시행착오를 겪었다.
- 하지만, 다시 문제를 읽고 어떤 목표를 가지고 데이터 결측치를 처리하는지를 생각해보니 결측치를 어떻게 처리할 지에 대한 판단 기준이 세워져 활용가능해졌다.

Try :

- 다음에는 RFM 점수화를 통해 고객을 명확하게 세그먼트로 나누고
  각 세그먼트 별로 특징과 비즈니스 전략을 연결하는 분석을 시도해보고 싶다.