고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

  EDA

```
SELECT *
FROM `secure-wonder-466601-q0.ModulabsRFMproject.data`
LIMIT 10
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGHT...	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 COAT HA...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WAT...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE HE...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT ...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

  EDA, table schema 확인 필요

```
SELECT *
FROM `secure-wonder-466601-q0.ModulabsRFMproject.data`
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC

페이지당 결과 수: 50 1 – 50 (전체 541909행)

데이터 수 세기

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

```
SELECT
  COUNT(InvoiceNo) AS I_N,
  COUNT(StockCode) AS S_C,
  COUNT(Description) AS D,
  COUNT(Quantity) AS Q,
  COUNT(InvoiceDate) AS I_D,
  COUNT(UnitPrice) AS U_P,
  COUNT(CustomerID) AS C_ID,
  COUNT(Country) AS C
FROM `secure-wonder-466601-q0.ModulabsRFMproject.data`
```

[결과 이미지를 넣어주세요]

행	I_N	S_C	D	Q	I_D	U_P	C_ID	C
1	541909	541909	540455	541909	541909	541909	406829	541909

EDA, 결측치 처리 필요

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 secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]

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

결측치 존재 항목 일부, CustomerID 항목에 편중

: 주요 항목에 대한 데이터 손실 불가피

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
WHERE StockCode = '85123A'
ORDER BY Description
```

[결과 이미지를 넣어주세요]

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

동일 항목임을 확인 가능한 추가 데이터(상품번호 등) 존재하는 경우

: 대표성을 가지는 데이터로 대체 / 데이터 통합해 대체 등

동일 항목임을 확인 불가한 경우

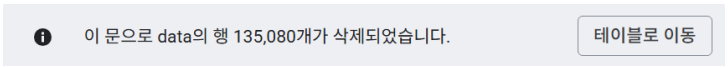
: 한 종류 데이터 제외 제거 / 모든 데이터 제거(데이터가 갖는 의미 관점에서 보다 정확)

결측치 처리

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

```
DELETE FROM secure-wonder-466601-q0.ModulabsRFMproject.data
WHERE Description IS NULL OR CustomerID IS NULL
```

[결과 이미지를 넣어주세요]



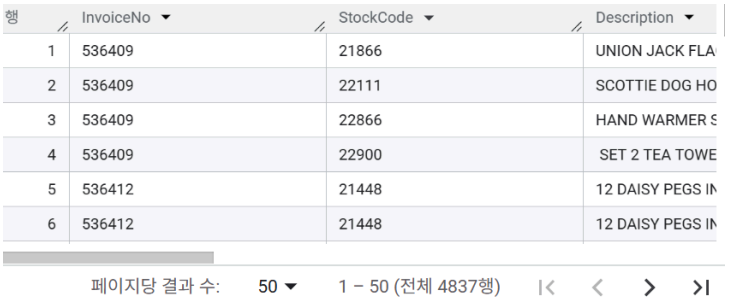
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT *
FROM
  secure-wonder-466601-q0.ModulabsRFMproject.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING COUNT(*)>1
ORDER BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
```

[결과 이미지를 넣어주세요]



행	InvoiceNo	StockCode	Description
1	536409	21866	UNION JACK FLA
2	536409	22111	SCOTTIE DOG HO
3	536409	22866	HAND WARMER S
4	536409	22900	SET 2 TEA TOWE
5	536412	21448	12 DAISY PEGS IN
6	536412	21448	12 DAISY PEGS IN

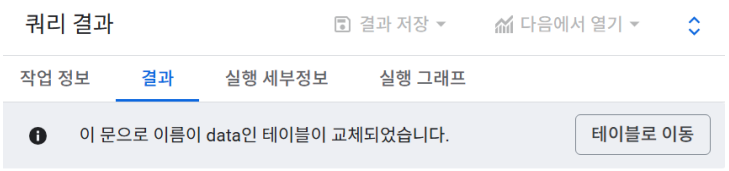
행의 모든 값이 동일한 데이터, 제거(데이터가 갖는 비중 조절, normalization)

중복값 처리

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

```
CREATE OR REPLACE TABLE secure-wonder-466601-q0.ModulabsRFMproject.data AS
SELECT DISTINCT *
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]



11-6. 데이터 전처리(3): 오류값 처리

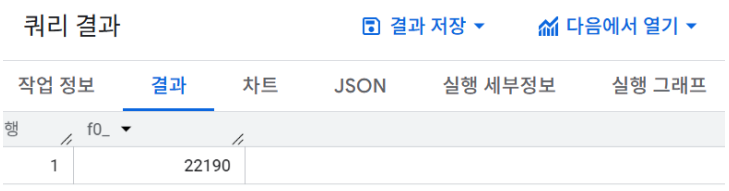
각 열의 데이터 탐색, 상황 및 데이터에 대한 배경 지식을 기반으로 전처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]



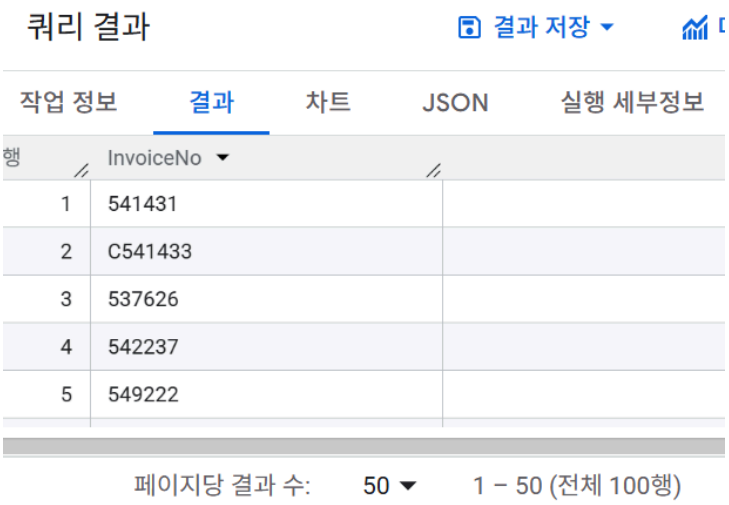
행	f0_
1	22190

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

EDA 위해 출력하는 head 개수는 전체 데이터에 대한 비중, 표본 추출 시 통계적 의미를 가지는 수준으로 조정될 필요가 있음

```
SELECT DISTINCT InvoiceNo
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
LIMIT 100
```

[결과 이미지를 넣어주세요]



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

  주문번호가 C로 시작하는 데이터와 price 혹은 quantity의 값이 음수인 데이터가 상호 대응되는지(필요충분) 확인 필요

  price 혹은 quantity가 0인 경우의 데이터 확인 필요

```
SELECT *
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100
```

[결과 이미지를 넣어주세요]

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

  취소 시 기존 주문번호 앞에 C가 추가되어 저장되는지,

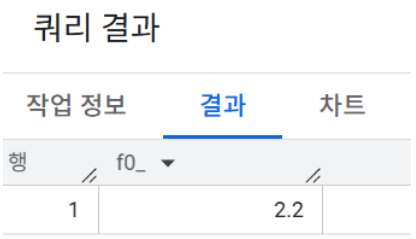
  그렇다면 취소되기 전 주문번호는 테이블에서 삭제되는지 확인하고,

  비율 계산 시 취소되기 전 주문번호는 전체 주문번호 수의 합에서 제외할 필요

```
SELECT
  ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/COUNT(InvoiceNo)*100, 1)
```

```
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]



StockCode 살펴보기

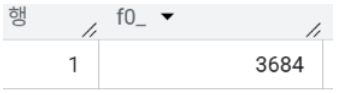
- **고유한 StockCode의 개수를 출력하기**

  마찬가지로 EDA 시 추출하는 표본의 개수는 통계적으로 유의한 수준이어야 함

  추출 횟수를 조절하는 것도 방법

```
SELECT COUNT(DISTINCT StockCode)
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]



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

    추출 표본 개수, 마찬가지로,,

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
GROUP BY StockCode
ORDER BY sell_cnt 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

- **StockCode의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값**들에는 어떤 코드들이 들어가 있는지 출력하기

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM secure-wonder-466601-q0.ModulabsRFMproject.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;

SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
)
WHERE number_count BETWEEN 0 AND 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

페이지당 결과 수: 50 1 – 8 (전체 8행)

- `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 outliers AS(
  SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
)
WHERE number_count BETWEEN 0 AND 1
)

SELECT
  ROUND(SAFE_DIVIDE(
    COUNTIF(StockCode IN (
      SELECT StockCode
      FROM outliers)), COUNT(StockCode)
  )*100, 2)
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

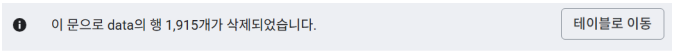
[결과 이미지를 넣어주세요]

행	f0_
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM secure-wonder-466601-q0.ModulabsRFMproject.data
WHERE StockCode IN
 (SELECT DISTINCT StockCode
 FROM
  (SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))
      AS number_count
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
  )
 WHERE number_count BETWEEN 0 AND 1
)
```

[결과 이미지를 넣어주세요]



이상 계산한 데이터를 이용해 데이터의 신뢰성을 가늠하고,

데이터 생성 과정을 개선?

Description 살펴보기

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

```
SELECT
  Description,
  COUNT(*) AS description_cnt
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30
```

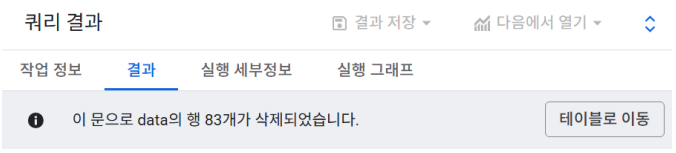
[결과 이미지를 넣어주세요]

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
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013
12	LUNCH BAG SPACEBOY DESIGN	1006

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
WHERE Description IN ('Next Day Carriage', 'High Resolution Image')
```

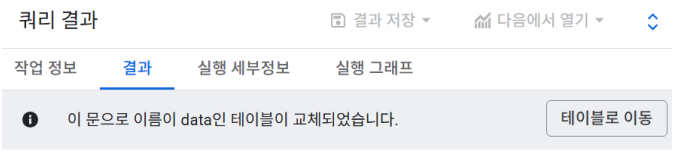
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE secure-wonder-466601-q0.ModulabsRFMproject.data AS
SELECT
  * EXCEPT(Description),
  UPPER(Description) AS Description
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]

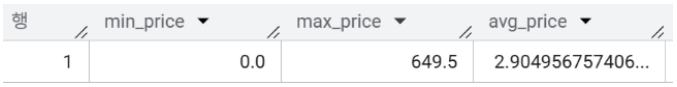


UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]

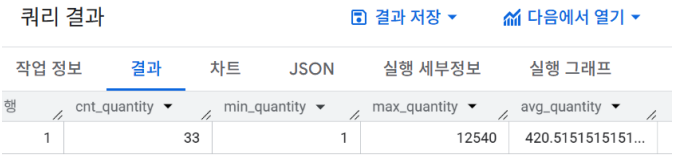


행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

- **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 secure-wonder-466601-q0.ModulabsRFMproject.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

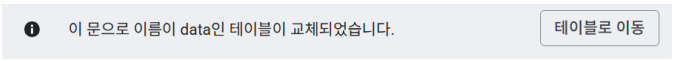


행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

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

```
CREATE OR REPLACE TABLE secure-wonder-466601-q0.ModulabsRFMproject.data AS
SELECT *
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
WHERE UnitPrice != 0
```

[결과 이미지를 넣어주세요]



번거롭지만 UnitPrice = 0인 경우에 대한 EDA를 통해 무료 상품, 행사 등의 경우 식별 시도?

나아가 timestamp 데이터를 추가해 이벤트 등이 미치는 영향 등 분석?

11-7. RFM 스코어

Recency

고객별로 각 구매일자 간 간격을 분석해 고객 분류 등에 사용도 가능

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

```
SELECT DATE(InvoiceDate) AS InvoiceDay,
  *
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceNo	StockCode
12	2010-12-07	537626	22805
13	2010-12-07	537626	22494
14	2010-12-07	537626	20780
15	2010-12-07	537626	84997D
16	2010-12-07	537626	21171
17	2010-12-07	537626	22773
18	2010-12-07	537626	84558A
19	2010-12-07	537626	22375
20	2010-12-07	537626	85167B
21	2010-12-07	537626	71477
22	2010-12-07	537626	22775

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  (SELECT MAX(DATE(InvoiceDate))
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
  ) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
```

[결과 이미지를 넣어주세요]

행	most_recent_date	InvoiceDay	InvoiceNo
1	2011-12-09	2011-01-18	541431
2	2011-12-09	2011-01-18	C541433
3	2011-12-09	2010-12-07	537626
4	2011-12-09	2010-12-07	537626
5	2011-12-09	2010-12-07	537626
6	2011-12-09	2010-12-07	537626
7	2011-12-09	2010-12-07	537626
8	2011-12-09	2010-12-07	537626

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

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

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
11	12357	2011-11-06

- **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 secure-wonder-466601-q0.ModulabsRFMproject.data
  GROUP BY CustomerID
)
```

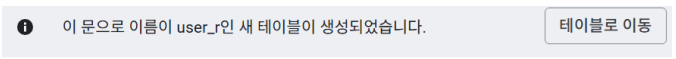
[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	12414	217
2	12658	18
3	12670	10
4	12971	3
5	13356	80
6	13448	16
7	13884	7
8	14030	18
9	14381	52
10	14627	311
11	14753	87

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

```
CREATE OR REPLACE TABLE secure-wonder-466601-q0.ModulabsRFMproject.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
  GROUP BY CustomerID
)
```

[결과 이미지를 넣어주세요]



Frequency

frequency 또한 일자 구간별로 빈도 나누어 계산하면 유의미할 것, 일자 구간을 나누는 것도 고객 특성을 고려 필요한지? 혹은 모든 고객에 대해 일관성있게 적용해야 할 지?

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84
7	12353	4
8	12354	58
9	12355	13
10	12356	58
11	12357	131
12	12358	17

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM secure-wonder-466601-q0.ModulabsRFMproject.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
10	12356	1573
11	12357	2708
12	12358	242

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

```
-- # 전체 거래 건수 계산, 구매한 아이템의 총 수량 계산 결과 테이블로 합쳐서 저장
CREATE OR REPLACE TABLE secure-wonder-466601-q0.ModulabsRFMproject.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
```

```
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM secure-wonder-466601-q0.ModulabsRFMproject.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 secure-wonder-466601-q0.ModulabsRFMproject.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency
1	16274	63	151	373
2	17968	81	151	373
3	13747	1	8	373
4	12791	1	96	373
5	15350	5	51	373
6	17643	8	71	373
7	14237	9	38	373
8	18074	13	190	373
9	13065	14	74	373
10	16583	14	111	373
11	14142	22	313	373
12	15165	27	160	373

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 2) AS user_total
FROM secure-wonder-466601-q0.ModulabsRFMproject.data
GROUP BY CustomerID
ORDER BY user_total DESC
```

[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	14646	278778.02
2	18102	259657.3
3	17450	189575.53
4	14911	128768.24
5	12415	123638.18
6	14156	113685.77
7	17511	88138.2
8	16684	65920.12
9	13694	62961.54
10	16029	60369.93

- **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 secure-wonder-466601-q0.ModulabsRFMproject.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  SAFE_DIVIDE(ut.user_total, purchase_cnt) AS user_average
FROM secure-wonder-466601-q0.ModulabsRFMproject.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 2) AS user_total
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

rchase_cnt	item_cnt	recency	user_total	user_average
2062	196556	1	278778.02	135.1978758486...
431	64124	0	259657.3	602.45313225058
339	69021	8	189575.53	559.2198525073...
5806	76823	1	128768.24	22.178477437134
774	76946	24	123638.18	159.7392506459...
1401	56896	9	113685.77	81.14615988579...
1074	63014	2	88138.2	82.06536312849...
280	49391	4	65920.12	235.4289999999...
581	61904	3	62961.54	108.3675387263...
260	33687	38	60369.93	232.1920384615...

RFM 통합 테이블 출력하기

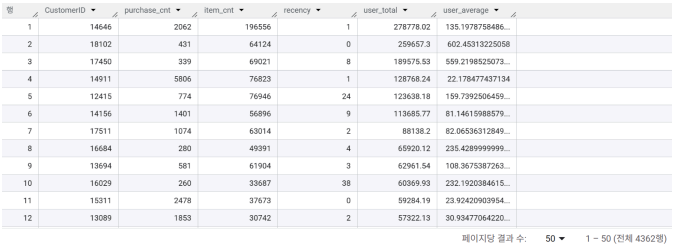
RFM 각 score에 따라 고객 분류(추후 예정),

RFM score 각각에 비중 할당해 계산?

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM secure-wonder-466601-q0.ModulabsRFMproject.user_rfm
ORDER BY user_total DESC
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	14646	2062	196556	1	278778.02	135.1978758486...
2	18102	431	64124	0	259657.3	602.45313225058
3	17450	339	69021	8	189575.53	559.2198525073...
4	14911	5806	76823	1	128768.24	22.178477437134
5	12415	774	76946	24	123638.18	159.7392506459...
6	14156	1401	56896	9	113685.77	81.14615988579...
7	17511	1074	63014	2	88138.2	82.06536312849...
8	16684	280	49391	4	65920.12	235.4289999999...
9	13694	581	61904	3	62961.54	108.3575387263...
10	16029	260	33687	38	60369.93	232.1920384615...
11	15311	2478	37673	0	59284.19	23.92420903954...
12	13089	1853	30742	2	57322.13	30.93477064220...

페이지당 결과 수: 50 1 – 50 (전체 4362행)

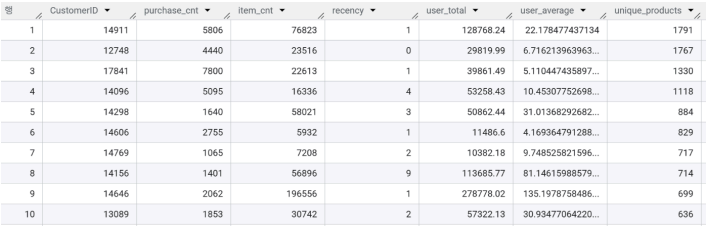
11-8. 추가 Feature 추출

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 secure-wonder-466601-q0.ModulabsRFMproject.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM secure-wonder-466601-q0.ModulabsRFMproject.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	14911	5806	76823	1	128768.24	22.178477437134	1791
2	12748	4440	23516	0	29819.99	6.716213963963...	1767
3	17841	7800	22613	1	39861.49	5.110447435897...	1330
4	14096	5095	16336	4	53258.43	10.45307752698...	1118
5	14298	1640	58021	3	50862.44	31.01368292682...	884
6	14606	2755	5932	1	11486.6	4.169364791288...	829
7	14769	1065	7208	2	10382.18	9.748525821596...	717
8	14156	1401	56896	9	113685.77	81.14615988579...	714
9	14646	2062	196556	1	278778.02	135.1978758486...	699
10	13089	1853	30742	2	57322.13	30.93477064220...	636

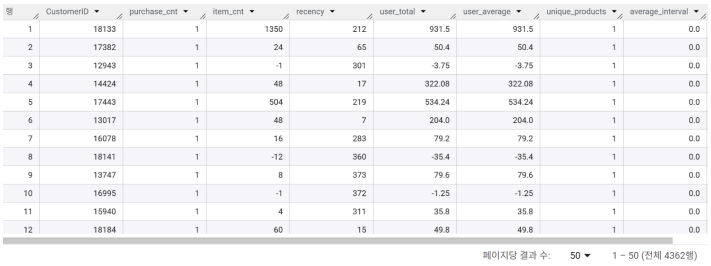
2. 평균 구매 주기

- **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 secure-wonder-466601-q0.ModulabsRFMproject.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
      secure-wonder-466601-q0.ModulabsRFMproject.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM secure-wonder-466601-q0.ModulabsRFMproject.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	18133	1	1350	212	931.5	931.5	1	0.0
2	17382	1	24	65	50.4	50.4	1	0.0
3	12943	1	-1	301	-3.75	-3.75	1	0.0
4	14424	1	48	17	322.08	322.08	1	0.0
5	17443	1	504	219	534.24	534.24	1	0.0
6	13017	1	48	7	204.0	204.0	1	0.0
7	16078	1	16	283	79.2	79.2	1	0.0
8	18141	1	-12	360	-35.4	-35.4	1	0.0
9	13747	1	8	373	79.6	79.6	1	0.0
10	16995	1	-1	372	-1.25	-1.25	1	0.0
11	15940	1	4	311	35.8	35.8	1	0.0
12	18184	1	60	15	49.8	49.8	1	0.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

3. 구매 취소 경향성

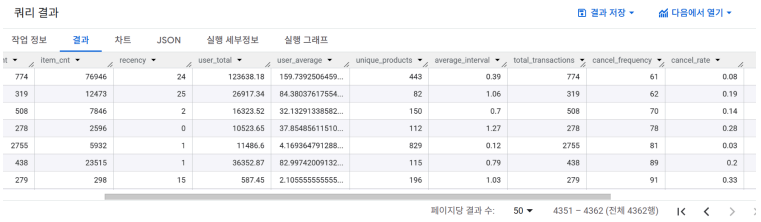
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE secure-wonder-466601-q0.ModulabsRFMproject.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    COUNTIF(InvoiceNo LIKE 'C%') AS cancel_frequency
  FROM secure-wonder-466601-q0.ModulabsRFMproject.data
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  ROUND(SAFE_DIVIDE(t.cancel_frequency, t.total_transactions), 2) AS cancel_rate
FROM `secure-wonder-466601-q0.ModulabsRFMproject.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
```

[결과 이미지를 넣어주세요]

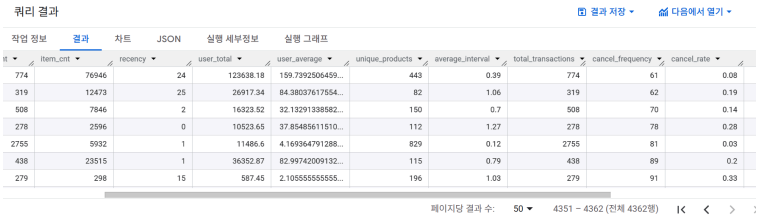


	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
774	76946	24	123638.18	159.7392506459...	443	0.39	774	61	0.08
319	12473	25	26917.34	84.38037617554...	82	1.06	319	62	0.19
508	7846	2	16323.52	32.13291338582...	150	0.7	508	70	0.14
278	2596	0	10523.65	37.85485611510...	112	1.27	278	78	0.28
2755	5932	1	11486.6	4.169364791288...	829	0.12	2755	81	0.03
438	23515	1	36352.87	82.99742009132...	115	0.79	438	89	0.2
279	298	15	587.45	2.105555555555...	196	1.03	279	91	0.33

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

```
SELECT *
FROM `secure-wonder-466601-q0.ModulabsRFMproject.user_data`
```

[결과 이미지를 넣어주세요]



	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
774	76946	24	123638.18	159.7392506459...	443	0.39	774	61	0.08
319	12473	25	26917.34	84.38037617554...	82	1.06	319	62	0.19
508	7846	2	16323.52	32.13291338582...	150	0.7	508	70	0.14
278	2596	0	10523.65	37.85485611510...	112	1.27	278	78	0.28
2755	5932	1	11486.6	4.169364791288...	829	0.12	2755	81	0.03
438	23515	1	36352.87	82.99742009132...	115	0.79	438	89	0.2
279	298	15	587.45	2.105555555555...	196	1.03	279	91	0.33

회고

[회고 내용을 작성해주세요]

Keep :

Problem : 사소한 문법 실수가 큰 시간 손실로 이어짐, 언어 구조 및 문법 정리와 연습 필요

Try : 다양한 함수 접해보기, 코드 길이 줄이고 가독성 높여보기