고객을 세그먼테이션하자 [프로젝트] 조연규

11-2. 데이터 불러오기

데이터 살펴보기

• 테이블에 있는 10개의 행만 출력하기

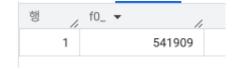
```
SELECT *
FROM `planar-sun-473602-n0.modulabs.data`
LIMIT 10;
```

[결과 이미지를 넣어주세요]

행 //	InvoiceNo ▼	StockCode ▼	Description ▼	Quantity •	InvoiceDate ▼	UnitPrice ▼	CustomerID ▼	Country ▼
1	536365	85123A	WHITE HANGING HEART T-LIG	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 H	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO	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
10	536367	84879	ASSORTED COLOUR BIRD ORN	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

• 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT
COUNT(*)
FROM
`planar-sun-473602-n0.modulabs.data`;
```



데이터 수 세기

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

```
SELECT
COUNT(*)
FROM
'planar-sun-473602-n0.modulabs.data';

SELECT
COUNT(InvoiceNo) AS InvoiceNo_count,
COUNT(StockCode) AS StockCode_count,
COUNT(Description) AS Description_count,
COUNT(Quantity) AS Quantity_count,
COUNT(InvoiceDate) AS InvoiceDate_count,
COUNT(UnitPrice) AS UnitPrice_count,
COUNT(CustomerID) AS CustomerID_count,
COUNT(Country) AS Country_count
FROM
'planar-sun-473602-n0.modulabs.data';
```

[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

```
#전데이터에서 누락 여부 확인
SELECT
 'InvoiceNo' AS column_name,
COUNT(*) - COUNT(InvoiceNo) AS missing_count,
 (COUNT(*) - COUNT(InvoiceNo)) * 100.0 / COUNT(*) AS missing_perc
entage
FROM
 `csproject-251001`.modulabs_01.data
UNION ALL
SELECT
 'StockCode' AS column_name,
COUNT(*) - COUNT(StockCode) AS missing_count,
(COUNT(*) - COUNT(StockCode)) * 100.0 / COUNT(*) AS missing_per
centage
FROM
 `csproject-251001`.modulabs_01.data
UNION ALL
SELECT
'Description' AS column_name,
COUNT(*) - COUNT(Description) AS missing_count,
 (COUNT(*) - COUNT(Description)) * 100.0 / COUNT(*) AS missing_pe
rcentage
FROM
`csproject-251001`.modulabs_01.data
UNION ALL
SELECT
 'Quantity' AS column_name,
 COUNT(*) - COUNT(Quantity) AS missing_count,
(COUNT(*) - COUNT(Quantity)) * 100.0 / COUNT(*) AS missing_perce
ntage
FROM
 `csproject-251001`.modulabs_01.data
UNION ALL
SELECT
'InvoiceDate' AS column_name,
COUNT(*) - COUNT(InvoiceDate) AS missing_count,
(COUNT(*) - COUNT(InvoiceDate)) * 100.0 / COUNT(*) AS missing_pe
rcentage
```

```
FROM
 `csproject-251001`.modulabs_01.data
UNION ALL
SELECT
'UnitPrice' AS column_name,
COUNT(*) - COUNT(UnitPrice) AS missing_count,
 (COUNT(*) - COUNT(UnitPrice)) * 100.0 / COUNT(*) AS missing_perc
entage
FROM
`csproject-251001`.modulabs_01.data
UNION ALL
SELECT
 'CustomerID' AS column_name,
COUNT(*) - COUNT(CustomerID) AS missing_count,
(COUNT(*) - COUNT(CustomerID)) * 100.0 / COUNT(*) AS missing_pe
rcentage
FROM
 `csproject-251001`.modulabs_01.data
UNION ALL
SELECT
 'Country' AS column_name,
COUNT(*) - COUNT(Country) AS missing_count,
(COUNT(*) - COUNT(Country)) * 100.0 / COUNT(*) AS missing_perce
ntage
FROM
 `csproject-251001`.modulabs_01.data;
SELECT
 'Description' AS column_name,
COUNT(*) - COUNT(Description) AS missing_count,
 ROUND((COUNT(*) - COUNT(Description)) * 100.0 / COUNT(*), 2) AS
missing_percentage
FROM
`csproject-251001`.modulabs_01.data
UNION ALL
SELECT
```

'CustomerID' AS column_name,

COUNT(*) - COUNT(CustomerID) AS missing_count,

ROUND((COUNT(*) - COUNT(CustomerID)) * 100.0 / COUNT(*), 2) AS missing_percentage

FROM

`csproject-251001`.modulabs_01.data;

[결과 이미지를 넣어주세요]

행 //	column_name ▼	missing_count -	missing_percenta
1	InvoiceNo	0	0.0
2	Country	0	0.0
3	UnitPrice	0	0.0
4	CustomerID	135080	24.92669433428
5	Quantity	0	0.0
6	Description	1454	0.268310731137
7	StockCode	0	0.0
8	InvoiceDate	0	0.0

행 //	column_name ▼	missing_count ▼ //	missing_percenta
1	CustomerID	135080	24.93
2	Description	1454	0.27

결측치 처리 전략

SELECT DISTINCT Description
FROM csproject-251001.modulabs_01.data
WHERE StockCode = '85123A'
AND Description IS NOT NULL;

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

 [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

결측치 처리

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

DELETE FROM `planar-sun-473602-n0.modulabs.data` WHERE CustomerID IS NULL OR Description IS NULL;

[결과 이미지를 넣어주세요]

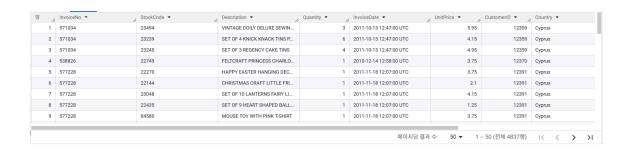
● 이 문으로 RFM_data의 행 135,080개가 삭제되었습니다.

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 `planar-sun-473602-n0.modulabs.RFM_data`
GROUP BY
InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,Cust omerID,Country
HAVING COUNT(*) > 1;



중복값 처리

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

CREATE OR REPLACE TABLE `planar-sun-473602-n0.modulabs.RFM_d ata` AS
SELECT DISTINCT

*
FROM
 `planar-sun-473602-n0.modulabs.RFM_data`;

[결과 이미지를 넣어주세요]

이 문으로 이름이 RFM_data인 테이블이 교체되었습니다.

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

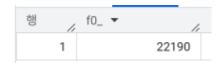
• 고유(unique)한 InvoiceNo 의 개수를 출력하기

SELECT
COUNT(DISTINCT InvoiceNo)

FROM

`planar-sun-473602-n0.modulabs.RFM_data`;

[결과 이미지를 넣어주세요]



• 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

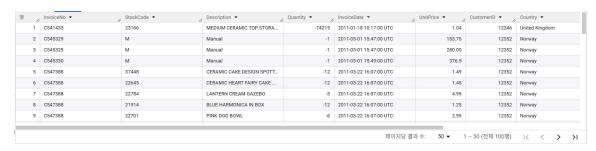
```
SELECT DISTINCT
InvoiceNo
FROM
`planar-sun-473602-n0.modulabs.RFM_data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]



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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```



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

```
SELECT
ROUND(SUM(
CASE
WHEN InvoiceNo LIKE 'C%' THEN 1
ELSE 0
END) / COUNT(*) * 100, 1)
FROM
`planar-sun-473602-n0.modulabs.RFM_data`;
```

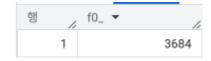
[결과 이미지를 넣어주세요]



StockCode 살펴보기

• 고유한 StockCode 의 개수를 출력하기

```
SELECT
COUNT(DISTINCT StockCode)
FROM
`planar-sun-473602-n0`.modulabs.RFM_data;
```



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

```
SELECT
StockCode,
COUNT(StockCode) AS cell_cnt
FROM
`planar-sun-473602-n0`.modulabs.RFM_data
GROUP BY StockCode
ORDER BY cell_cnt DESC
LIMIT 10;
```

행 //	StockCode ▼	cell_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 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
WITH UniqueStockCodes AS (
SELECT DISTINCT StockCode
FROM `planar-sun-473602-n0.modulabs.RFM_data`
)
```

SELECT

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

[결과 이미지를 넣어주세요]

행 //	number_count	• /	stock_cnt ▼	//
1		5		3676
2		0		7
3		1		1

• 제품과 관련되지 않은 거래 기록을 제거하기

DELETE `planar-sun-473602-n0`.modulabs.RFM_data
WHERE
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1;

[결과 이미지를 넣어주세요]

● 이 문으로 RFM_data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

• 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

SELECT
Description,
COUNT(*) AS description_cnt

FROM

`planar-sun-473602-n0`.modulabs.RFM_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

• 서비스 관련 정보를 포함하는 행들을 제거하기

DELETE `planar-sun-473602-n0`.modulabs.RFM_data WHERE

Description LIKE '%POSTAGE%' OR Description LIKE '%CARRIAGE%' OR Description LIKE '%BANK CHARGES%' OR

Description LIKE '%DISCOUNT%' OR Description LIKE '%ADJUSTME NT%' OR Description LIKE '%SAMPLES%' OR Description LIKE '%AMA ZON FEE%' OR

Description LIKE '%CRACKED%';

[결과 이미지를 넣어주세요]

● 이 문으로 RFM_data의 행 86개가 삭제되었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

CREATE OR REPLACE TABLE project_name.modulabs_project.data AS SELECT

* EXCEPT (Description),

UPPER(Dercription) AS Description

FROM project_name.modulabs_project.data;

[결과 이미지를 넣어주세요]

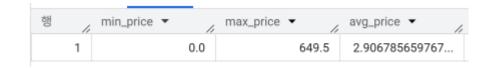
● 이 문으로 이름이 RFM_data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

• UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```
SELECT
MIN(UnitPrice) AS min_price,
MAX(UnitPrice) AS max_price,
AVG(UnitPrice) AS avg_price
FROM
`planar-sun-473602-n0`.modulabs.RFM_data;
;
```

[결과 이미지를 넣어주세요]

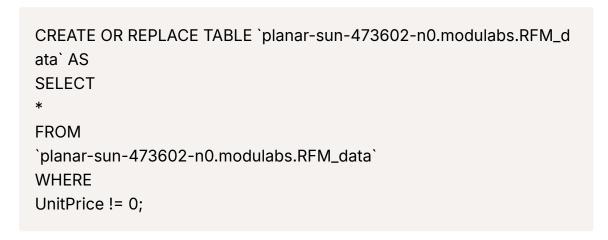


• 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기

```
SELECT
COUNT(*) AS cnt_quantity,
MIN(Quantity) AS min_quantity,
MAX(Quantity) AS max_quantity,
AVG(Quantity) AS avg_quantity
FROM
`planar-sun-473602-n0`.modulabs.RFM_data
WHERE
UnitPrice = 0;
```



• UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기



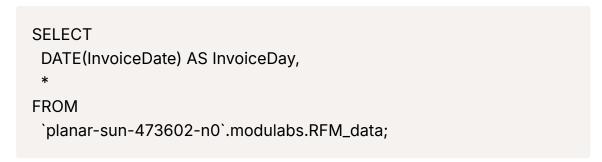
[결과 이미지를 넣어주세요]

이 문으로 이름이 RFM_data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

• InvoiceDate 컬럼을 연월일 자료형으로 변경하기



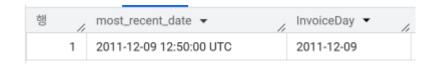
[결과 이미지를 넣어주세요]



가장 최근 구매 일자를 MAX() 함수로 찾아보기

SELECT
MAX(InvoiceDate) AS most_recent_date,
DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM
`planar-sun-473602-n0`.modulabs.RFM_data;

[결과 이미지를 넣어주세요]



• 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS most_recent_purchase_date
FROM
`planar-sun-473602-n0`.modulabs.RFM_data
```

```
GROUP BY CustomerID;
```



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

```
SELECT
CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS rece
ncy
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM project_name.modulabs_project.data
GROUP BY CustomerID
);
```



• 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 user_r 이라는 이름의 테이블로 저장하기

CREATE OR REPLACE TABLE `planar-sun-473602-n0`.modulabs.user_r
AS
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS most_recent_purchase_date
FROM
`planar-sun-473602-n0`.modulabs.RFM_data
GROUP BY CustomerID;

[결과 이미지를 넣어주세요]

● 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

• 고객마다 고유한 InvoiceNo의 수를 세어보기

SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM
`planar-sun-473602-n0`.modulabs.RFM_data
GROUP BY CustomerID;



• 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
CustomerID,
SUM(Quantity) AS total_quantity_purchased
FROM
`planar-sun-473602-n0`.modulabs.RFM_data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]



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

```
CREATE OR REPLACE TABLE `planar-sun-473602-n0`.modulabs.user_r f AS
WITH

purchase_cnt AS (

SELECT

CustomerID,

COUNT(DISTINCT InvoiceNo) AS purchase_cnt

FROM

`planar-sun-473602-n0`.modulabs.RFM_data

GROUP BY CustomerID
),

item_cnt AS (

SELECT

CustomerID,
```

```
SUM(Quantity) AS item_cnt
  FROM
   `planar-sun-473602-n0`.modulabs.RFM_data
  GROUP BY CustomerID
)
SELECT
 pc.CustomerID,
pc.purchase_cnt,
ic.item_cnt,
ur.most_recent_purchase_date AS recency
FROM
 purchase_cnt AS pc
JOIN
item cnt AS ic
ON pc.CustomerID = ic.CustomerID
JOIN
 `planar-sun-473602-n0`.modulabs.user_r AS ur
 ON pc.CustomerID = ur.CustomerID;
```

이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Monetary

• 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
CustomerID,
ROUND(SUM(Quantity * UnitPrice)) AS user_total
FROM
`planar-sun-473602-n0`.modulabs.RFM_data
GROUP BY CustomerID;
```



• 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 'planar-sun-473602-n0'.modulabs.user_r
fm AS
SELECT
 rf.CustomerID AS CustomerID,
 rf.purchase_cnt,
 rf.item_cnt,
 rf.recency,
 ut.user_total,
 ROUND(ut.user_total / rf.purchase_cnt, 2) AS user_average
FROM
 `planar-sun-473602-n0`.modulabs.user_rf AS rf
 LEFT JOIN
  SELECT
   CustomerID,
   ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM
   `planar-sun-473602-n0`.modulabs.RFM_data
  GROUP BY CustomerID
 ) AS ut
 ON rf.CustomerID = ut.CustomerID;
```

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

• 최종 user_rfm 테이블을 출력하기

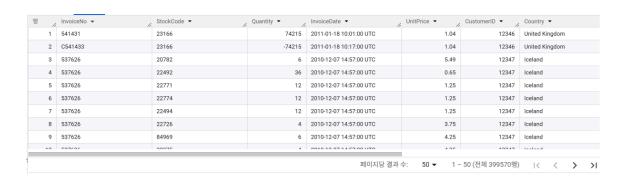
SELECT

*

FROM

`planar-sun-473602-n0`.modulabs.user_rfm;

[결과 이미지를 넣어주세요]



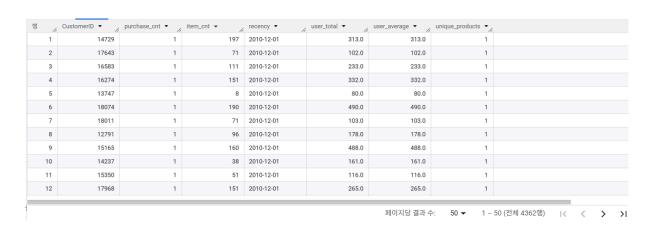
11-8. 추가 Feature 추출

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 `planar-sun-473602-n0`.modulabs.user_data AS WITH unique_products AS (

```
SELECT
CustomerID,
COUNT(DISTINCT CustomerID) AS unique_products
FROM
    `planar-sun-473602-n0`.modulabs.RFM_data
GROUP BY CustomerID
)
SELECT
ur.*,
up.* EXCEPT (CustomerID)
FROM
    `planar-sun-473602-n0`.modulabs.user_rfm AS ur
JOIN
unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 평균 구매 소요 일수를 계산하고, 그 결과를 user_data 에 통합

```
CREATE OR REPLACE TABLE `planar-sun-473602-n0`.modulabs.user_data
AS
WITH
purchase_intervals AS (
SELECT
```

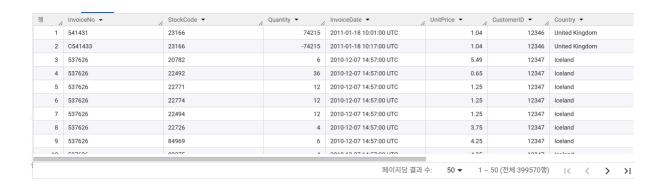
```
CustomerID,
   CASE
    WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0
    ELSE ROUND(AVG(interval_), 2)
   END AS average_interval
  FROM
   (
    SELECT
     CustomerID,
     DATE_DIFF(recency, LAG(recency) OVER (PARTITION BY CustomerID
       ORDER BY recency), DAY) AS interval_
    FROM
     `planar-sun-473602-n0`.modulabs.RFM_data
    WHERE
     CustomerID IS NOT NULL
   )
  GROUP BY CustomerID
)
SELECT
u.*,
 pi.* EXCEPT (CustomerID)
FROM
 `planar-sun-473602-n0`.modulabs.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_products ▼//	average_interval 🕶	
1	17968	1	151	2010-12-01	265.0	265.0	1	0.0	
2	13065	1	74	2010-12-01	206.0	206.0	1	0.0	
3	15350	1	51	2010-12-01	116.0	116.0	1	0.0	
4	18074	1	190	2010-12-01	490.0	490.0	1	0.0	
5	14142	1	313	2010-12-01	312.0	312.0	1	0.0	
6	14237	1	38	2010-12-01	161.0	161.0	1	0.0	
7	16583	1	111	2010-12-01	233.0	233.0	1	0.0	
8	18011	1	71	2010-12-01	103.0	103.0	1	0.0	
9	13747	1	8	2010-12-01	80.0	80.0	1	0.0	
10	17908	1	169	2010-12-01	232.0	232.0	1	0.0	
페이지당결과 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 'planar-sun-473602-n0'.modulabs.user_data
AS
WITH
TransactionInfo AS (
  SELECT
   CustomerID,
   COUNT(DISTINCT purchase_cnt) AS total_transactions,
   COUNT(
    CASE
     WHEN purchase_cnt LIKE 'C%' THEN 1
    END) AS cancel_frequency
  FROM
   `planar-sun-473602-n0`.modulabs.RFM_data
  GROUP BY CustomerID
)
SELECT
u.*,
t.* EXCEPT (CustomerID),
 ROUND(IFNULL(t.cancel_frequency * 100.0 / t.total_transactions, 0), 2) AS
cancel_rate
FROM
`planar-sun-473602-n0`.modulabs.user_data AS u
LEFT JOIN
TransactionInfo AS t
 ON u.CustomerID = t.CustomerID;
```

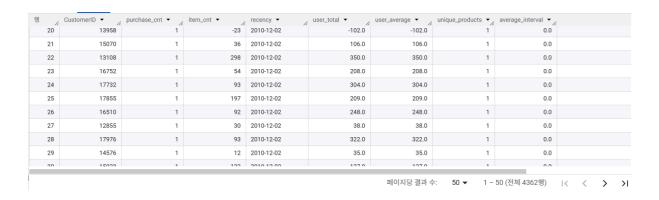


다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 user_data 를 출력하기

SELECT * FROM

`planar-sun-473602-n0`.modulabs.user_data;

[결과 이미지를 넣어주세요]



회고

[회고 내용을 작성해주세요]

Keep: 어렴풋이 겨우겨우 끝가지는 해 봤지만 데이터를 신뢰하고 보스에게 보고할 정도 믿음이 없다.

- Problem :마지막 Frequency 지점에서 문제가 발생한 것으로 예측됨
- 읽고 따라는 했지만 RFM 프로세스에 대한 이해도 빈약으로 결과가 어긋남

● 프로세스를 따라 주석을 달고 문제가 될만한 부분에 주석 처리 못해 복기가 어려웠음

Try: 쿼리문 작성에 대한 해석이 더 필요하고 직접 장성이 어려운 쿼리문을 어떻게 읽고 해석하가에 따라 결과 차이가 크게난다.

행 //	CustomerID ▼	purchase_cnt ▼	item_cnt ▼	recency ▼	user_total ▼	user_average ▼ //	unique_products ▼//			
1	14729	1	197	2010-12-01	313.0	313.0	1			
2	17643	1	71	2010-12-01	102.0	102.0	1			
3	16583	1	111	2010-12-01	233.0	233.0	1			
4	16274	1	151	2010-12-01	332.0	332.0	1			
5	13747	1	8	2010-12-01	80.0	80.0	1			
6	18074	1	190	2010-12-01	490.0	490.0	1			
7	18011	1	71	2010-12-01	103.0	103.0	1			
8	12791	1	96	2010-12-01	178.0	178.0	1			
9	15165	1	160	2010-12-01	488.0	488.0	1			
10	14237	1	38	2010-12-01	161.0	161.0	1			
11	15350	1	51	2010-12-01	116.0	116.0	1			
12	17968	1	151	2010-12-01	265.0	265.0	1			
						페이지당 결과 수	±: 50 ▼ 1 −	50 (전체 4362행) <	<	>