

노드 11



11-7. RFM 스텝부터 진행 과정에서 결과 캡처를 하였습니다. 이전 스텝은 전처리 이후에 캡처하여 결과가 정답과 다릅니다.

11-4. 결측치 제거

컬럼 별 누락된 값의 비율 계산

```
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS ratio
FROM `innate-concept-464902-r1.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 ratio
FROM `innate-concept-464902-r1.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 ratio
FROM `innate-concept-464902-r1.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 ratio
FROM `innate-concept-464902-r1.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 ratio
FROM `innate-concept-464902-r1.modulabs_project.data`
```

```

UNION ALL
SELECT
    'UnitPrice' AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS missing_percentag
FROM `innate-concept-464902-r1.modulabs_project.data`
UNION ALL
SELECT
    'CustomerID' AS column_name,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS missing_percentag
FROM `innate-concept-464902-r1.modulabs_project.data`
UNION ALL
SELECT
    'Country' AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS missing_percentag
FROM `innate-concept-464902-r1.modulabs_project.data`

```

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

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

페이지당 결과 수: 50

1 - 8 (전체 8행)

결측치 처리 전략

```
SELECT Distinct StockCode, Description
FROM `innate-concept-464902-r1.modulabs_project.data`
WHERE StockCode = '85123A'
```

결측치 처리

```
DELETE FROM `innate-concept-464902-r1.modulabs_project.data`
WHERE CustomerID IS NULL
OR Description IS NULL
```

11-5. 중복값 처리

중복값 확인

```
SELECT *, COUNT(*) AS cnt
FROM `innate-concept-464902-r1.modulabs_project.data`
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice
-- InvoiceNo + StockCode + Description + ... + Country 이 모두 동일한 행들끼리
-- 즉, 각 컬럼의 데이터 값이 완전히 같은 행을 묶는 행위.
HAVING cnt > 1
```

중복값 처리

```
CREATE OR REPLACE TABLE??
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.data`
SELECT DISTINCT *
FROM `innate-concept-464902-r1.modulabs_project.data`;
```

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceNo	StockCode	Quantity	UnitPrice	CustomerID	Country	Description	InvoiceDate	most_recent_date
1	541431	23166	74215	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...	2011-01-18	2011-01-18
2	C541433	23166	-74215	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...	2011-01-18	2011-01-18
3	537626	851678	30	1.25	12347	Iceland	BLACK GRAND BAROQUE PHOT...	2010-12-07	2011-12-07
4	537626	22805	12	1.25	12347	Iceland	BLUE DRAWER KNOB ACRYLIC ...	2010-12-07	2011-12-07
5	537626	85232D	3	4.95	12347	Iceland	SET/3 DECOUPAGE STACKING ...	2010-12-07	2011-12-07
6	537626	22775	12	1.25	12347	Iceland	PURPLE DRAWERKNOB ACRYLI...	2010-12-07	2011-12-07
7	537626	20780	12	4.65	12347	Iceland	BLACK EAR MUFF HEADPHONES	2010-12-07	2011-12-07
8	537626	20782	6	5.49	12347	Iceland	CAMOUFLAGE EAR MUFF HEA...	2010-12-07	2011-12-07
9	537626	84969	6	4.25	12347	Iceland	BOX OF 6 ASSORTED COLOUR T...	2010-12-07	2011-12-07
10	537626	84558A	24	2.95	12347	Iceland	3D DOG PICTURE PLAYING CAR...	2010-12-07	2011-12-07
11	537626	22195	12	1.65	12347	Iceland	LARGE HEART MEASURING SP...	2010-12-07	2011-12-07
12	537626	22497	4	4.25	12347	Iceland	SET OF 2 TINS VINTAGE BATHR...	2010-12-07	2011-12-07

11-6. 오류값 처리

InvoiceNo 살펴보기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM `innate-concept-464902-r1.modulabs_project.data`
```

행	f0_
1	21784

1. 취소한 거래 확인하기

```
SELECT *
FROM `innate-concept-464902-r1.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100
```

2. 취소된 거래 비율 확인하기

```
SELECT
  ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*))
FROM `innate-concept-464902-r1.modulabs_project.data`
```

⇒ 2.2%

StockCode 살펴보기

```
SELECT COUNT(DISTINCT StockCode)
FROM `innate-concept-464902-r1.modulabs_project.data`
```

1. StockCode 별 등장 빈도 확인하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `innate-concept-464902-r1.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

쿼리 결과			
작업 정보 <u>결과</u> 차트 JSON 실행 세부정보 실행 그i			
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	22197	1110	
9	23203	1108	
10	20727	1099	

2. StockCode 의 문자열 내 숫자의 길이를

(전처리가 다 된 상태에서 결과 캡처를 하여 결과 값이 정답과 다릅니다.)

```
WITH UniqueStockCodes AS (  
  SELECT DISTINCT StockCode  
  FROM project_name.modulabs_project.data  
)  
SELECT  
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) /  
  COUNT(*) AS stock_cnt  
FROM UniqueStockCodes  
GROUP BY number_count  
ORDER BY stock_cnt DESC;
```

쿼리 결과				
작업 정보		결과	차트	JSON
행	number_count	stock_cnt		
1	5	3674		

3. 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지 확인

```
SELECT DISTINCT StockCode, number_count  
FROM (  
  SELECT StockCode,  
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))  
  FROM `innate-concept-464902-r1.modulabs_project.data`  
)  
WHERE number_count < 2
```

3-1. 해당 코드 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트??

```
SELECT ROUND(SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_
FROM `innate-concept-464902-r1.modulabs_project.data`
```

⇒ 0.48%

4. 제품과 관련되지 않은 거래기록 제거

```
DELETE FROM `innate-concept-464902-r1.modulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM `innate-concept-464902-r1.modulabs_project.data`
  WHERE
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))
)
```

Description 살펴보기

```
SELECT Description, COUNT(*) AS description_cnt
FROM `innate-concept-464902-r1.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

1. 대소문자가 혼합된 Description이 있는지 확인

```
SELECT DISTINCT Description
FROM `innate-concept-464902-r1.modulabs_project.data`
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

2.

서비스 관련 정보를 포함하는 행들을 제거

```
DELETE
FROM `innate-concept-464902-r1.modulabs_project.data`
```

```
WHERE Description IN ('Next Day Carriage', 'High Resolution Image')
```

3.

대소문자를 혼합하고 있는 데이터를 대문자로 표준화

```
CREATE OR REPLACE TABLE innate-concept-464902-r1.modulabs_project.da
SELECT * EXCEPT (Description), UPPER(TRIM(Description)) AS Description
FROM `innate-concept-464902-r1.modulabs_project.data`
```

UnitPrice 살펴보기

1. **UnitPrice**의 최솟값, 최댓값, 평균

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(Uni
FROM `innate-concept-464902-r1.modulabs_project.data`
```

쿼리 결과				
작업 정보 결과 차트 JSON 실행 세부정보				
행	min_price	max_price	AVG_price	
1	0.0	649.5	2.904956757406...	

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

```
SELECT COUNT(UnitPrice) AS unitprice, MIN(Quantity) AS min_quantity, MAX
FROM `innate-concept-464902-r1.modulabs_project.data`
WHERE UnitPrice = 0
```


쿼리 결과						결과 저장 ▼	
작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프	
행	unitprice ▼	min_quantity ▼	max_quantity ▼	AVG_quantity ▼			
1	33	1	12540	420.5151515151...			

3. UnitPrice = 0 를 제거

```
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.data`
SELECT *
FROM `innate-concept-464902-r1.modulabs_project.data`
WHERE UnitPrice = 0
```

11-7. RFM 스코어

RFM 스코어 구하기

Recency (구매 최신성)

- 고객이 마지막으로 구매한 시점을 나타냄.
- 최근에 구매한 고객들은 더 자주 구매할 가능성이 높기 때문에, 최신성 점수가 높는지 고려해야함.
- '마지막 구매일로부터 현재까지 경과한 일수' 중요

1. InvoiceDate를 'YYYY-MM-DD' 형태로 변경

```
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.data`
SELECT
  * EXCEPT(InvoiceDate),
  DATE(InvoiceDate) AS InvoiceDate
FROM `innate-concept-464902-r1.modulabs_project.data`;
```

InvoiceDate ▼	
2010-12-01	
2010-12-01	
2010-12-03	
2010-12-05	
2010-12-06	
2010-12-08	
2010-12-09	
2010-12-09	
2010-12-09	
2010-12-09	

2. 가장 최근 구매 일자를 MAX() 함수로 찾기

```
SELECT
  MAX(InvoiceDate) OVER() AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `innate-concept-464902-r1.modulabs_project.data`
```

행	most_recent_date	InvoiceDay	li
1	2011-12-09	2011-10-07	5
2	2011-12-09	2011-01-19	5
3	2011-12-09	2011-10-07	5
4	2011-12-09	2011-06-08	5
5	2011-12-09	2011-09-30	5
6	2011-12-09	2011-08-18	5
7	2011-12-09	2011-07-31	5
8	2011-12-09	2011-04-27	5
9	2011-12-09	2011-10-20	5
10	2011-12-09	2011-11-17	5
11	2011-12-09	2011-10-30	5
12	2011-12-09	2011-09-27	5
13	2011-12-09	2011-11-17	5
14	2011-12-09	2011-07-07	5

3. 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장

```
ALTER TABLE `innate-concept-464902-r1.modulabs_project.data`
ADD COLUMN most_recent_date DATE
```

```
UPDATE `innate-concept-464902-r1.modulabs_project.data` AS t
SET most_recent_date = sub.most_recent_date
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS most_recent_date
  FROM `innate-concept-464902-r1.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
) AS sub
WHERE t.CustomerID = sub.CustomerID;
```

업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

CustomerID ▼	Country ▼	Description ▼	InvoiceDate ▼	most_recent_date ▼
12346	United Kingdom	MEDIUM CERAMIC TOP STORA...	2011-01-18	2011-01-18
12346	United Kingdom	MEDIUM CERAMIC TOP STORA...	2011-01-18	2011-01-18
12347	Iceland	ALARM CLOCK BAKELIKE CHO...	2010-12-07	2011-12-07
12347	Iceland	FOUR HOOK WHITE LOVEBIRDS	2010-12-07	2011-12-07
12347	Iceland	RED DRAWER KNOB ACRYLIC E...	2010-12-07	2011-12-07
12347	Iceland	RED TOADSTOOL LED NIGHT LI...	2010-12-07	2011-12-07
12347	Iceland	BLACK EAR MUFF HEADPHONES	2010-12-07	2011-12-07
12347	Iceland	CAMOUFLAGE EAR MUFF HEA...	2010-12-07	2011-12-07
12347	Iceland	BOX OF 6 ASSORTED COLOUR T...	2010-12-07	2011-12-07
12347	Iceland	MINI PAINT SET VINTAGE	2010-12-07	2011-12-07
12347	Iceland	EMERGENCY FIRST AID TIN	2010-12-07	2011-12-07
12347	Iceland	RED 3 PIECE RETROSPOT CUTL...	2010-12-07	2011-12-07
12347	Iceland	BLUE 3 PIECE POLKADOT CUTL...	2010-12-07	2011-12-07
12347	Iceland	BLACK GRAND BAROQUE PHOT...	2010-12-07	2011-12-07

4. 가장 최근 일자(last_purchase_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산

```

SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `innate-concept-464902-r1.modulabs_project.data`
  GROUP BY CustomerID
);

```

쿼리 결과		
작업 정보	결과	차트
JSON		
행	CustomerID	recency
1	12420	63
2	12919	8
3	12924	88
4	13016	73
5	13193	61
6	13357	257
7	13782	12
8	13787	75
9	14148	233
10	14191	1
11	14431	298
12	14447	18
13	14452	10
14	14495	50

5. 고객별 Recency 계산 – 결과를 modulabs_project.user_r 테이블로 저장

```

DROP TABLE IF EXISTS `innate-concept-464902-r1.modulabs_project.user_r`;

-- 고객별 Recency 계산 뒤 user_r 테이블 생성
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.user_r`
WITH
  per_user AS (      -- ① 고객별 마지막 구매일
    SELECT
      CustomerID,
      MAX(InvoiceDate) AS last_purchase_date  -- TIMESTAMP → DATE
    FROM `innate-concept-464902-r1.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
    GROUP BY CustomerID
  ),
  global_max AS (    -- ② 데이터 전체에서 가장 최근 날짜
    SELECT MAX(InvoiceDate) AS most_recent_date
    FROM `innate-concept-464902-r1.modulabs_project.data`
  )
SELECT                -- ③ Recency(일수) 계산
  u.CustomerID,

```

```
DATE_DIFF(g.most_recent_date, u.last_purchase_date, DAY) AS recency
FROM per_user AS u
CROSS JOIN global_max AS g;
```

작업 정보	결과	차트	JSON	설정
행	CustomerID	recency		
1	16446	0		
2	14397	0		
3	14441	0		
4	12526	0		
5	17389	0		
6	13069	0		
7	12423	0		
8	12662	0		
9	12985	0		
10	16626	0		
11	16954	0		
12	14446	0		
13	15910	0		
14	12680	0		
--	-----	-		

Frequency (구매 빈도)

- Frequency를 계산하는 단계에서는 고객의 구매 빈도 또는 참여 빈도에 초점
- 예를 들어 한 명의 고객이 구매를 2번 했는데 각각 아이템을 4개씩 구매한 경우, 해당 고객의 거래 건수는 2회겠지만 실제로 구매한 수량은 8개가 됩니다. 이 두가지 측면을 모두 포착하기 위해 두 개를 모두 계산

1. 전체 거래 건수 계산

```
SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM innate-concept-464902-r1.modulabs_project.data
GROUP BY CustomerID
```

쿼리 결과		
<div> 작업 정보 결과 차트 JSON </div>		
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
10	12356	3
11	12357	1
12	12358	2
13	12359	6
14	12360	3

2. 구매한 아이템의 총 수량 계산

```
SELECT CustomerID, sum(Quantity) AS item_cnt
FROM innate-concept-464902-r1.modulabs_project.data
GROUP BY CustomerID
```

쿼리 결과

작업 정보	결과	차트	JSON
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	
13	12359	1599	
14	12360	1156	
--	-----	--	

3. (1) 거래 건수, (2) 구매 수량, (3) recency를 통합해 user_rf 생성

```
-- (1) 거래 건수, (2) 구매 수량, (3) recency를 통합해 user_rf 생성
DROP TABLE IF EXISTS `innate-concept-464902-r1.modulabs_project.user_rf`

-- 통합 테이블 생성
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.us
WITH
  -- (1) 고객별 거래 건수
  purchase_cnt AS (
    SELECT
      CustomerID,
      COUNT(DISTINCT InvoiceNo) AS purchase_cnt
    FROM `innate-concept-464902-r1.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
    GROUP BY CustomerID
```



```

),

-- (2) 고객별 총 구매 수량
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM `innate-concept-464902-r1.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

-- (3) JOIN은 여기 SELECT 안에서!
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 `innate-concept-464902-r1.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt	item_cnt	recency	
1	12713	1	505	0	
2	14569	1	79	1	
3	13298	1	96	1	
4	15520	1	314	1	
5	13436	1	76	1	
6	15195	1	1404	2	
7	14204	1	72	2	
8	15471	1	256	2	
9	12650	1	250	3	
10	17914	1	457	3	
11	16528	1	171	3	
12	14578	1	240	3	
13	16569	1	93	3	
14	12478	1	233	3	

Monetary (구매 가치)

- 고객이 지불한 총 금액에 초점
- 이 때 총 지출액을 계산할수도 있고, 거래당 평균 거래 금액을 계산
예를 들어, 한 명의 고객이 총 2번의 구매를 했고, 그 합산 금액이 10만원인 경우, 총 지출액은 10만원, 거래당 평균 거래 금액은 5만원이 되는 것

1. 고객별 총 지출액 계산

```
SELECT CustomerID, ROUND(SUM(Quantity * UnitPrice), 1) AS price
FROM `innate-concept-464902-r1.modulabs_project.data`
GROUP BY CustomerID
```

쿼리 결과

작업 정보 **결과** 차트 JSON

행	CustomerID	price
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.6
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7
12	12358	928.1
13	12359	6183.0
14	12360	2302.1
--	-----	----

2. 고객별 평균 거래 금액 계산

```
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.us
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(ut.user_total / rf.purchase_cnt, 1) AS user_average
FROM `innate-concept-464902-r1.modulabs_project.user_rf` AS rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total,
  FROM `innate-concept-464902-r1.modulabs_project.data`
```

```
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

RFM 통합 테이블 출력하기

```
select *
from `innate-concept-464902-r1.modulabs_project.user_rfm`
```

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	13298	1	96	1	360.0	360.0
3	13436	1	76	1	197.0	197.0
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	12442	1	181	3	144.0	144.0
10	12650	1	250	3	242.0	242.0
11	16528	1	171	3	244.0	244.0
12	16569	1	93	3	124.0	124.0
13	15992	1	17	3	42.0	42.0
14	15318	1	642	3	313.0	313.0
...

Windows

정품 인증

[실정]으로 이동하여 Windows를 정품 인증합니다.

페이지당 결과 수:

50

1 - 50 (전체 4363행)

<<

<

>

>>

11-8. 추가 Featuer 추출

RFM 이외의 유저별 구매 패턴 추출하기 (RFM에 추가추가 하는 과정이었음)

- RFM 분석 방법은 Recency, Frequency, Monetary에 의해 고객을 세그먼트이션하는 방법이지만 허점 존재
사이트에 방문한 횟수가 동일하고 비슷한 금액을 지출했지만 구매 패턴이 다른 사람을 분류하지 못함.

<다양한 측면에서 데이터 분석하기> ⇒ 추후, 클러스터링 및 분석 결과 시각화 하기

1. 구매하는 제품의 다양성
2. 평균 구매 주기
3. 구매 취소 경향성

클러스터링 알고리즘이란?

- 비슷한 특성을 가진 데이터 포인트들을 그룹화하는 기술
- 데이터 속에 숨겨진 구조나 패턴을 찾아서 비슷한 데이터들끼리 그룹을 지어주는 것

1. 구매하는 제품의 다양성

- 1) 고객 별로 구매한 상품들의 고유한 수를 계산 (제품 종류의 수를 세는 것)
- 2) `user_rfm` 테이블과 결과를 합치고
- 3) `user_data` 라는 이름의 테이블에 저장

```
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.us
WITH unique_products AS (
  SELECT CustomerID, COUNT(DISTINCT StockCode) AS unique_products
  FROM `innate-concept-464902-r1.modulabs_project.data`
  GROUP BY CustomerID
)

SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `innate-concept-464902-r1.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID
```

쿼리 결과

작업 정보		결과	차트	JSON	실행 세부정보		실행 그래프	
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	
1	13135	1	4300	196	3096.0	3096.0	1	
2	17948	1	144	147	359.0	359.0	1	
3	12814	1	48	101	86.0	86.0	1	
4	16138	1	-1	368	-8.0	-8.0	1	
5	17763	1	12	263	15.0	15.0	1	
6	13829	1	-12	359	-102.0	-102.0	1	
7	14351	1	12	164	51.0	51.0	1	
8	15524	1	4	24	440.0	440.0	1	
9	16257	1	1	176	22.0	22.0	1	
10	17347	1	216	86	229.0	229.0	1	
11	17382	1	24	65	50.0	50.0	1	
12	13099	1	288	99	207.0	207.0	1	
13	13391	1	4	203	60.0	60.0	1	
14	13120	1	12	238	31.0	31.0	1	
15	17307	1	-144	365	-153.0	-153.0	1	
16	15313	1	25	110	52.0	52.0	1	
17	16797	1	288	59	418.0	418.0	1	

2. 평균 구매 주기

- 목표: 고객들의 쇼핑 패턴을 이해하는 것
(여기서는 고객 별 재방문 주기를 확인할 것)
- HOW? 고객들의 구매와 구매 사이의 기간이 평균적으로 몇인지 확인.
⇒ 고객의 다음 구매일이 예측 됨.
- **CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS**
average_interval
ROUND(AVG(interval_), 2) IS NULL
은 특정 고객이 단 하나의 구매 건만 있어서, '바로 직전 구매일'이 없는 경우를 의미. 이 경우 '평균 소요 일수'를 0을 저장.
CASE 절의 ELSE 조건에서는
interval_의 평균을 계산하여 **average_interval**에 저장.

```
CREATE OR REPLACE TABLE project_name.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
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
```

```

SELECT
  CustomerID,
  DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID
FROM
  project_name.modulabs_project.data
WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

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

```

쿼리 결과									
작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프			
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	
1	15195	1	1404	2	3861.0	3861.0	1	0.0	
2	13366	1	144	50	56.0	56.0	1	0.0	
3	15070	1	36	372	106.0	106.0	1	0.0	
4	12943	1	-1	301	-4.0	-4.0	1	0.0	
5	14705	1	100	198	179.0	179.0	1	0.0	
6	16061	1	-1	269	-30.0	-30.0	1	0.0	
7	12603	1	56	21	613.0	613.0	1	0.0	
8	15510	1	2	330	250.0	250.0	1	0.0	
9	13307	1	4	120	15.0	15.0	1	0.0	
10	16454	1	2	64	6.0	6.0	1	0.0	
11	17291	1	72	308	551.0	551.0	1	0.0	
12	17923	1	50	282	208.0	208.0	1	0.0	
13	13185	1	12	267	71.0	71.0	1	0.0	
14	17443	1	504	219	534.0	534.0	1	0.0	
15	13302	1	5	155	64.0	64.0	1	0.0	
16	17715	1	384	200	326.0	326.0	1	0.0	
17	17763	1	12	263	15.0	15.0	1	0.0	

3. 구매 취소 경향성

이 단계에서는 고객의 취소 패턴을 깊게 파고 본다.

1. 취소 빈도(cancle_frequency)

- 고객 별로 취소한 거래의 총 횟수
- 거래 취소 가능성이 높은 고객 식별 가능

- 취소 빈도는 불만족의 정도이거나, 다른 문제에 대한 지표일 수 있다.
따라서, 취소 빈도를 이해함으로써 거래 취소 횟수를 줄이고 고객 만족도를 높이는 전략을 세울 수 있다.

2. 취소 비율(cancel_rate)

- 각 고객의 한 모든 거래 중에서 취소를 한 거래의 비율.
- 특정 고객이 원래 취소를 잘 하는 고객인지 확인하기 위한 지표.
- 이를 바탕으로 취소 비율 감소를 위해 어떤 고객 대상군을 공략해야할지 전략을 세울 수 있다.

```
CREATE OR REPLACE TABLE `innate-concept-464902-r1.modulabs_project.us

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT CASE WHEN LEFT(InvoiceNo, 1) = 'C' THEN InvoiceNo ELSE InvoiceNo END) AS cancelled_transactions,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `innate-concept-464902-r1.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
),

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(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), 1) AS interval_
    FROM `innate-concept-464902-r1.modulabs_project.data`
  )
)
```



```

WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

SELECT
    u.CustomerID,
    u.purchase_cnt,
    u.item_cnt,
    u.recency,
    u.user_total,
    u.user_average,
    t.unique_products,
    p.average_interval,
    t.total_transactions,
    t.cancel_frequency,
    SAFE_DIVIDE(t.cancel_frequency, t.total_transactions) AS cancel_rate
FROM `innate-concept-464902-r1.modulabs_project.user_rfm` AS u
LEFT JOIN TransactionInfo AS t
    ON u.CustomerID = t.CustomerID
LEFT JOIN purchase_intervals AS p
    ON u.CustomerID = p.CustomerID;

```

쿼리 결과

작업 정보결과차트JSON실행 세부정보실행 그래프

행	CustomerID ▾	purchase_cnt ▾	item_cnt ▾	recency ▾	user_total ▾	user_average ▾	unique_products ▾	average_interval ▾	total_transactions ▾	cancel_frequency ▾	cancel_rate ▾
1	15619	1	136	10	336.0	336.0	3	0.0	1	0	0.0
2	17458	1	170	15	317.0	317.0	52	0.0	1	0	0.0
3	14675	1	336	16	596.0	596.0	93	0.0	1	0	0.0
4	17985	1	171	22	631.0	631.0	20	0.0	1	0	0.0
5	16071	1	180	44	326.0	326.0	21	0.0	1	0	0.0
6	15592	1	354	46	389.0	389.0	24	0.0	1	0	0.0
7	15947	1	898	82	1708.0	1708.0	29	0.0	1	0	0.0
8	13663	1	91	179	189.0	189.0	13	0.0	1	0	0.0
9	13339	1	326	200	860.0	860.0	54	0.0	1	0	0.0