5일차 RFM 부분 과제

5일차 RFM 부분 과제

프로젝트의 목적

> RFM 분석을 통해 분석하고 있는 서비스의 현황(AS-IS) 파악하기

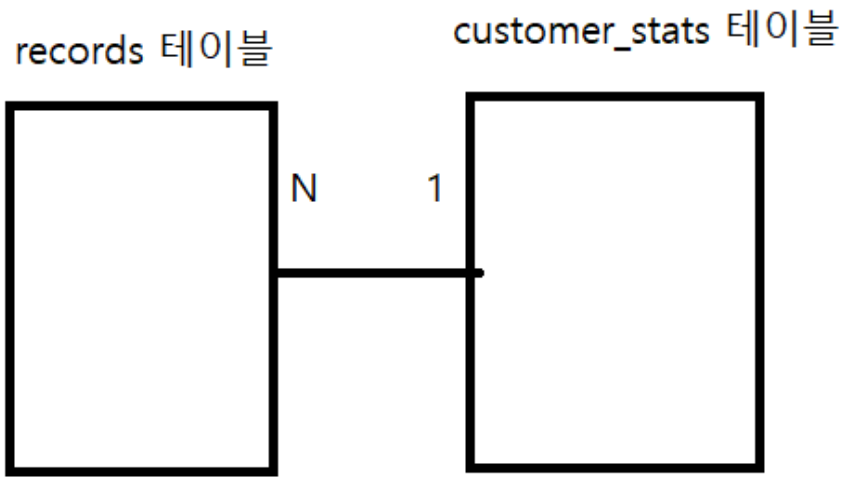
목차 구성

> 데이터 설명
>
> - US E-Commerce Records 2020 데이터 셋을 설명
>
> EDA
>
> - 유저들의 데이터가 어떻게 들어 있는가? (records 테이블과 customer_stats 테이블 분석)
> - 각 유저들의 Recency, Frequency, Monetary 값 산출
> - Recency, Frequency, Monetary 각 항목의 평균값, 최대, 최솟값을 구해봅니다.
> - 전체 유저의 Recency, Frequency, Monetary의 최솟값, 최댓값을 기준으로 N개의 구간을 나누어 각 구간에 포함되는 유저들의 숫자를 구해보고, 어느 구간에 유저들이 많이 몰려있는지 확인해봅니다. 각 항목의 구간을 몇개로 나눌지 고민해보세요.

데이터 설명

US E-Commerce Records 2020 데이터 셋은 2020년 미국 전자 상거래의 주문 거래 데이터를 모음한 데이터 셋이다. 자세한 내용은 이곳을 누르면 상세히 볼 수 있다. 해당 내용을 기반으로 하여 총 두개의 테이블로 구성되어 있으며, 고객이 물품을 주문 및 판매 데이터가 기록 된 records 테이블과 고객 관련 데이터가 모인 customer_stats 테이블 두가지가 존재한다.

각각의 테이블은 customer_id로 테이블이 연결되어 있으며 orders 다 대 customer_stats 일의 구조를 띈다. 그림으로 간략하게 나타내면 다음과 같이 그릴 수 있겠다.



EDA

records 테이블과 customer_stats 테이블의 분석

records 테이블

캐글에 있는 자료와 solvesql의 자료등을 토대로 각 컬럼을 분석한 결과 각 컬럼은 다음과 같이 간략하게 정의 내릴 수 있다.

order_date : 주문 날짜 2020-01-01 ~ 2020-12-30 일까지의 date 자료형이 있음

order_id : 주문 아이디. 한명의 고객이 다수의 물품을 구매했다면 records 테이블에 다수의 레코드가 등록이 된다. 예를들어 customer_id가 JM-15250인 고객이 의자 5개, 형광등 4개 등을 구입했다고 했을 때 다음과 같이 레코드가 등록이 된다.

omer_id	segment	country	city	state	postal_code	region	product_id	category	sub_category	product_name
15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-ST-10002743	Office Supplies	Storage	SAFCO Boltless Steel Shelving
15250	Consumer	United States	Huntsville	Texas	77340	Central	FUR-FU-10002116	Furniture	Furnishings	Tenex Carpeted, Granite-Look or Clear
15250	Consumer	United States	Huntsville	Texas	77340	Central	FUR-CH-10003199	Furniture	Chairs	Office Star - Contemporary Task Swivel
15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-AR-10003158	Office Supplies	Art	Fluorescent Highlighters by Dixon
15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-BI-10000301	Office Supplies	Binders	GBC Instant Report Kit
15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-BI-10000343	Office Supplies	Binders	Pressboard Covers with Storage Hooks
15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-AP-10004708	Office Supplies	Appliances	Fellowes Superior 10 Outlet Split Surge
15250	Consumer	United States	Baltimore	Maryland	21215	East	FUR-CH-10004218	Furniture	Chairs	Global Fabric Manager's Chair, Dark G

ship_mode : 배송 등급에 관한 컬럼이다. Standard Class, First Class 등이 있다.

customer_id : 고객의 id를 가리킨다.

segment : 고객의 타입을 나타낸다. consumer, corporate 등이 있다.

Country : 고객의 국가를 알려주는 컬럼

city : 고객의 도시를 알려주는 컬럼

state : 고객의 주를 알려주는 컬럼

postal_code : 고객의 주소 중 우편번호를 알려주는 컬럼

region : 고객이 위치한 지역을 나타냄

product_id : 제품의 아이디를 나타냄. category(3)-subcategory(2)-고유아이디 순으로 제품 아이디가 결정된다.

category : 제품의 1차 분류 카테고리를 나타냄

sub_category : 제품의 1차 분류 카테고리를 기반하여 상세 카테고리를 나타냄

product_name : 고객이 구매한 제품의 이름을 나타냄

sales : 판매가를 나타냄. 만약 의자 4개를 25달러에 샀다면 4개의 총 sales가 25달러라는 뜻이다.

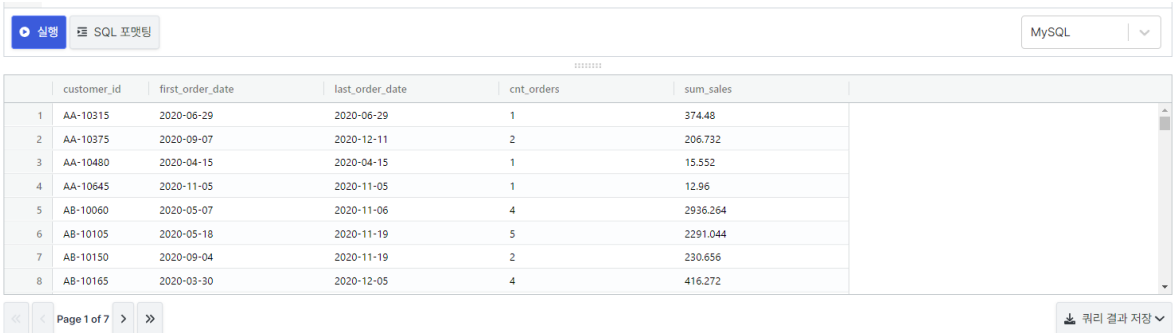
quantity : 주문 수량을 나타냄

discount : 할인율을 나타냄

profit : 매출에 대비한 순 이익을 나타냄 +가 될 수도 -가 될 수도 있다.

총 18개의 컬럼이 있는 것을 알 수 있다.

customer_stats 테이블



고객의 정보를 가진 테이블이다. 다음과 같이 총 5개의 칼럼이 있다.

customer_id : 고객의 id

first_order_date : 처음 주문한 날짜

last_order_date : 마지막으로 주문한 날짜

cnt_orders : 주문횟수 (**중요! records 테이블의 주문 id의 갯수를 기반으로 카운트 한 컬럼이다.**)

sum_sales : 해당 고객이 총 구매한 금액을 가리킨다.

각 유저들의 Recency, Frequency, Monetary 값 산출

	customer_id	retency	frequency	monetary
1	AA-10315	2020-06-29	1	374.48
2	AA-10375	2020-12-11	2	206.732
3	AA-10480	2020-04-15	1	15.552
4	AA-10645	2020-11-05	1	12.96
5	AB-10060	2020-11-06	4	2936.264
6	AB-10105	2020-11-19	5	2291.044
7	AB-10150	2020-11-19	2	230.656
8	AB-10165	2020-12-05	4	416.272

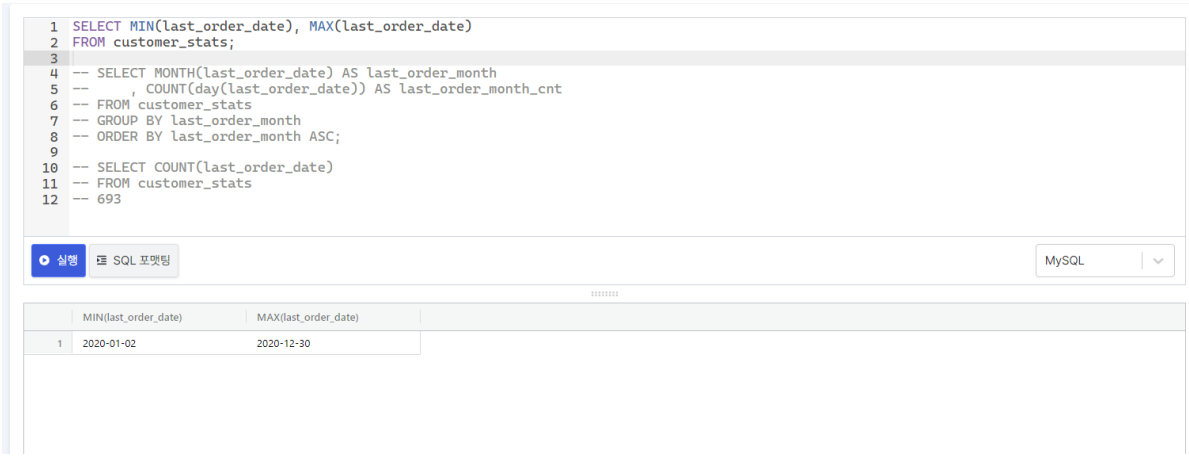
Page 1 of 7

customer_stats 테이블로 간단하게 해결할 수 있었다.

last_order_date를 recency로 cnt_orders를 frequency로 sum_sales를 monetary로 별칭을 붙여 테이블을 재 구성함.

Recency, Frequency, Monetary 각 항목의 평균값, 최대, 최솟값 및 구간 산출

Recency

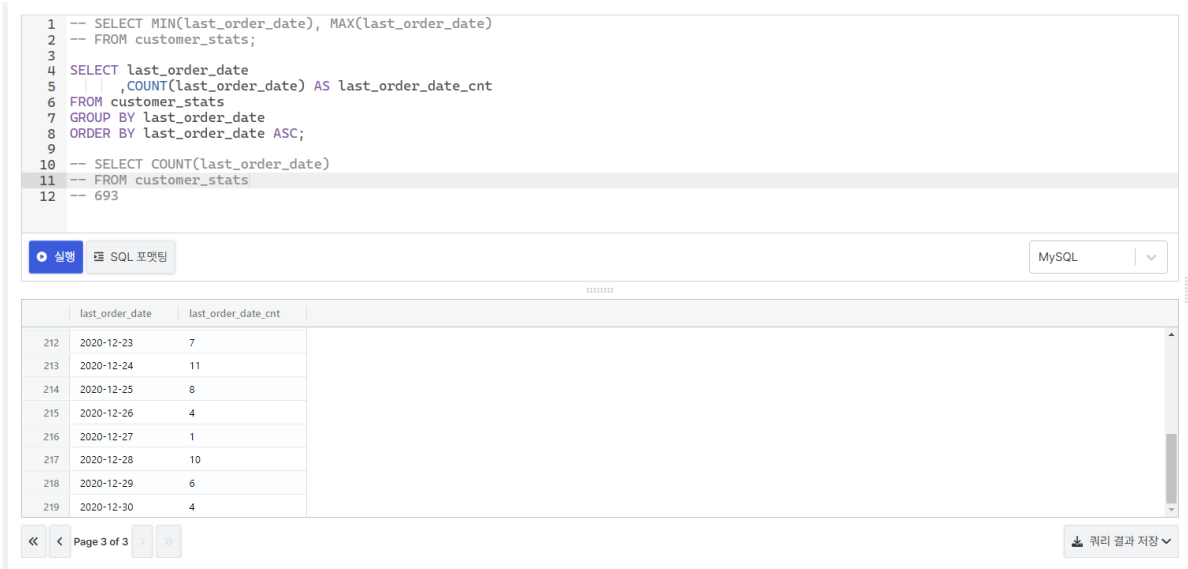


Recency가 오래 된 일자 : **20.01.02**

Recency 가장 최근 일자 : **20.12.30**

일자 기준으로 중위 수 구하기

last_order_date를 카운트하여 일자별 중위 수를 구함.



고객 데이터 전체 693개 중 11월 3일이 중위수에 가장 근접한 결과가 나옴.

월 별로 그룹으로 묶어 월 별 일자를 카운트하는 쿼리도 작성을 해봄

```sql
SELECT MONTH(last_order_date) AS last_order_month
     ,COUNT(DAY(last_order_date)) AS last_order_day_cnt
FROM customer_stats
GROUP BY last_order_month
ORDER BY last_order_month ASC;

-- SELECT COUNT(last_order_date)
-- FROM customer_stats
-- 693
```

실행 | SQL 포맷팅 | MySQL

	last_order_month	last_order_day_cnt
1	1	13
2	2	4
3	3	18
4	4	11
5	5	21
6	6	32
7	7	34
8	8	29

Page 1 of 1 | 쿼리 결과 저장

11월, 12월 전체 693개 데이터 중 357개로 Recency인 최근 주문 일자가 가장 최근 일자인 20.12.30과 비교하여 2개월 이내에 주문한 회원이 전체의 50% 이상 된다는 것도 알 수 있다.

Frequency

```sql
SELECT MIN(cnt_orders)
     ,MAX(cnt_orders)
     ,AVG(cnt_orders)
FROM customer_stats;
```

실행 | SQL 포맷팅 | MySQL

	MIN(cnt_orders)	MAX(cnt_orders)	AVG(cnt_orders)
1	1	8	2.4343

Page 1 of 1 | 쿼리 결과 저장

frequency에 해당하는 cnt_orders(주문 횟수)의 최소, 최대, 평균 값을 도출함

최소 값 : 1

최대 값 : 8

평균 : 2.43

```sql
SELECT cnt_orders
     ,  COUNT(customer_id)
FROM customer_stats
GROUP BY cnt_orders
ORDER BY cnt_orders DESC;

-- SELECT MIN(cnt_orders)
--        ,MAX(cnt_orders)
--        ,AVG(cnt_orders)
```

실행 | SQL 포맷팅 | MySQL

	cnt_orders	COUNT(customer_id)
1	8	2
2	7	2
3	6	9
4	5	39
5	4	85
6	3	156
7	2	200
8	1	200

Page 1 of 1 | 쿼리 결과 저장

```
SELECT cnt_orders
     , COUNT(customer_id) AS cnt_orders_customers_cnt
FROM customer_stats
GROUP BY cnt_orders
ORDER BY cnt_orders DESC;

-- SELECT MIN(cnt_orders)
--        ,MAX(cnt_orders)
--        ,AVG(cnt_orders)
```

실행 | SQL 포맷팅 | MySQL

	cnt_orders	cnt_orders_customers_cnt
1	8	2
2	7	2
3	6	9
4	5	39
5	4	85
6	3	156
7	2	200
8	1	200

Page 1 of 1 | 쿼리 결과 저장

이어서 cnt_orders를 그룹화 하여 cnt_orders를 기준으로 고객들의 갯수를 도출하는 쿼리도 작성 해봄
그 결과 주문 횟수가 1, 2회가 전체 고객의 50% 이상인 것을 알 수 있었음.

Monetary

```
SELECT MIN(sum_sales)
     ,MAX(sum_sales)
     ,ROUND(AVG(sum_sales), 3)
FROM customer_stats;
```

실행 | SQL 포맷팅 | MySQL

	MIN(sum_sales)	MAX(sum_sales)	ROUND(AVG(sum_sales), 3)
1	1.188	14203.278	1058.031

Page 1 of 1 | 쿼리 결과 저장

Monetary에 해당하는 sum_sales(매출 합계, 즉 고객이 구매한 총 금액)의 최소, 최대, 평균 값을 도출함

최소 값 : 1.188

최대 값 : 14203.278

평균 : 1058.031

```
SELECT *
FROM customer_stats
    INNER JOIN records ON
    customer_stats.customer_id = records.customer_id
ORDER BY sum_sales
```

실행 | SQL 포맷팅 | MySQL

...de	region	product_id	category	sub_category	product_name	sales	quantity	discount	profit
	West	OFF-BI-10002813	Office Supplies	Binders	Avery Reinforcements for Hole-Punch Pages	1.188	2	0.7	-0.99
	South	OFF-BI-10000145	Office Supplies	Binders	Zipper Ring Binder Pockets	2.808	3	0.7	-1.9656
	Central	FUR-FU-10000206	Furniture	Furnishings	GE General Purpose, Extra Long Life, Showcase & Floodlight Incandescent Bulbs	2.91	1	0	1.3677
	West	OFF-FA-10003472	Office Supplies	Fasteners	Bagged Rubber Bands	3.024	3	0.2	-0.6048
	West	OFF-AR-10002335	Office Supplies	Art	DIXON Oriole Pencils	5.16	2	0	1.3416
	East	OFF-LA-10003388	Office Supplies	Labels	Avery 5	5.76	2	0	2.8224
	West	OFF-AR-10002053	Office Supplies	Art	Premium Writing Pencils, Soft, #2 by Central Association for the Blind	5.96	2	0	1.6688
	Central	OFF-FA-10000611	Office Supplies	Fasteners	Binder Clips by OIC	2.368	2	0.2	0.8288

Page 1 of 34 | 쿼리 결과 저장

어떠한 것을 깨달았고 앞으로의 문제는?

1. **'United States E-Commerce records 2020' 데이터 셋의 원래의 로우 테이블은 records 테이블 하나였지만 데이터리안 측에서 감사하게도 RFM 분석을 수월하게 하고자 customer_stats 테이블을 만들어 프로젝트를 쉽게 수행하였다. 하지만 역량 상승을 위해 raw_table을 기반으로 RFM 분석 테이블을 구현하는 연습도 해야 될 것이다.**

1. **RFM 을 각각 어떠한 기준으로 잡아야 하는지 아직 깨닫지 못했다. 다양한 EDA를 시도해봤지만 어떤 기준으로 해야할지 잘 모르겠다. 팀원 분들은 어떻게 했는지 발표나 조언을 들어야 깨닫을 수 있는 기회가 생길듯?**

1. **뜬금없지만 서브쿼리에 대한 중요성을 깨달았다. 내가 근본적으로 원하고자 하는 테이블을 완벽히 만들지 못하여 엑셀로 쿼리 결과 저장을 하여 엑셀 함수를 이용해 결과를 내거나 계산기로 두들겨서 결과를 냈다. 서브쿼리를 이용하면 내가 근본적으로 원하고자 하는 검색 테이블을 제대로 구현할 수 있지 않을까?**