가설2 (etc)

작성일시	@2021년 7월 5일 오후 7:44

♉ 목차

▼ 목차 (삼각형을 클릭해주세요)

- ♉ 목차
- ♉사용 DataFrame
  - (1) Order_info
  - (2) Orders.csv
  - (3) products.csv
- ♉ Data 전처리
  - (0) Orders DataFrame 결측치 확인/ 처리
  - (1) days_since_prior_order에 대해 세부적으로 확인하고 시각화
  - (2) Orders DataFrame 이상치 확인/ 처리
- ♉ 가설 검증
  - 1 . 주말(토,일)에 주문량이 더 많다
    - (1) groupby수행하여 요약변수 생성 (Total_orders)
    - (2) order_dow를 요일 이름으로 변환하여 파생변수 생성 (day)
    - (3) 시각화
  - 2. 요일 & 시간별 주문량 알아보기
    - (1) 요일, 시간별 주문량을 구하기 위해 groupby 수행
    - (2) order_dow를 요일 이름으로 변환하는 파생변수 추가
    - (3) 요일/시간별 주문량을 pivot table로 만들어 확인
    - (4) Visualize
  - 3. 첫 주문 / 재주문 일 때 많이 구입한 상품은?
    - (1) orders DataFrame에서 첫 주문/재주문 내역 추출
    - (2) 첫 주문일 때 많이 구입한 상품 best 20
    - 2-1) first_order & order_info DataFrame을 merge
    - 2-2) groupby 수행하여 요약변수 생성 (order_cnt)
    - 2-3) 상품명을 알기 위해 product DataFrame과 Merge 수행

2-4) 주문량 상위 20개의 데이터 추출 및 시각화
✔️ 분석 결과
(3) 재 주문일 때 많이 구입한 상품 best 20
3-1) not_first_order & order_info DataFrame을 merge
3-2) groupby 수행하여 요약변수 생성 (order_cnt)
3-3) 상품명을 알기 위해 product DataFrame과 Merge 수행
3-4) 주문량 상위 20개의 데이터 추출 및 시각화
✔️ 분석 결과
4. 첫주문일때 주문 상품의 갯수보다 재주문일때 주문 상품의 갯수에 차이가 없다
(1) 첫주문 내역에서 1개의 주문 내역 당 판매하는 물건의 갯수
1-1) 데이터 준비
1-2) groupby 를 수행하여 요약변수 생성 (product_number)
(2) 재주문 내역에서 1개의 주문 내역 당 판매하는 물건의 갯수
2-1) 데이터 준비
2-2) groupby 를 수행하여 요약변수 생성 (product_number)
(3) 첫주문 내역과 재주문 내역에서 판매량에 대한 등분산 검정 실시
✔️ 분석 결과
(4) 독립표본 t검정 결과
✔️ 분석 결과

♉사용 DataFrame

(1) Order_info

- order_products__prior.csv 파일과 order_products__train.csv 파일을 merge한 데이터
- 어떤 상품을 구매했고, 상품을 장바구니에 담은 순서, 재구매한 상품인지에 대한 정보
- 32434489 rows , 4 columns 으로 구성

Aa column 명	기능 설명	type	NaN값 여부
order_id	주문 ID	int64	X
product_id	주문한 상품ID	int64	X
add_to_cart_order	장바구니에 담은 순서	int64	X
reordered	재구매한 상품인지 아닌지 표현 (1: 재구매 , 0: 최초 구매)	int64	X

(2) Orders.csv

- 주문 정보, 마지막 주문으로부터 얼마나 걸렸는지에 대한 정보
- 3421083 rows , 7 columns 로 구성

Aa column 명	기능 설명	type	NaN값 여부
order_id	주문 ID	int64	X
user_id	유저 ID	int64	X
eval_set	prior : 과거 구매한 주문내역인 경우 train , test : 최근 구매한 주문내역인 경우	object	X
order_number	구매 순서 (몇번째 구매인지)	int64	X
order_dow	구매 요일 (0: Sunday ~ 6: Saturday)	int64	X
order_hour_of_day	구매 시간	int64	X
days_since_prior_order	마지막 구매일로부터 걸린 시간(단위 : 일, NA : 첫구매)	float64	206209개

(3) products.csv

- product_id에 대한 정보
- 49688 rows , 4 columns 로 구성

Aa column 명	기능 설명	type	NaN 값 여부
product_id	order_products__prior.csv, order_products__train.csv 의 product_id와 연결할 수 있는 Join Key	int64	X
product_name	상품 이름	object	X
aisle_id	상품 세부 카테고리 ID	int64	X
department_id	상품 카테고리 ID	int64	X

➰ Data 전처리

(0) Orders DataFrame 결측치 확인/ 처리

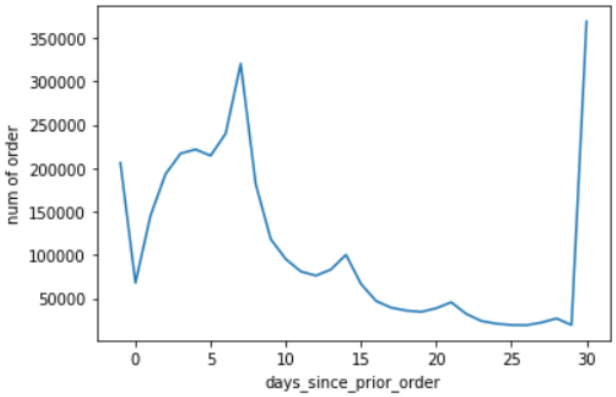
```
order_id                      0
user_id                       0
eval_set                      0
order_number                  0
order_dow                     0
order_hour_of_day             0
days_since_prior_order    206209
```

▶ Orders의 null값 갯수 확인 결과

- 마지막 구매로부터 걸린 시간인 'days_since_prior_order' 에 206209개의 결측치가 존재함을 확인
- 이 column에 존재하는 NA값은 첫 주문임을 의미함
  - 그러므로 결측값을 -1로 대체

(1) days_since_prior_order에 대해 세부적으로 확인하고 시각화

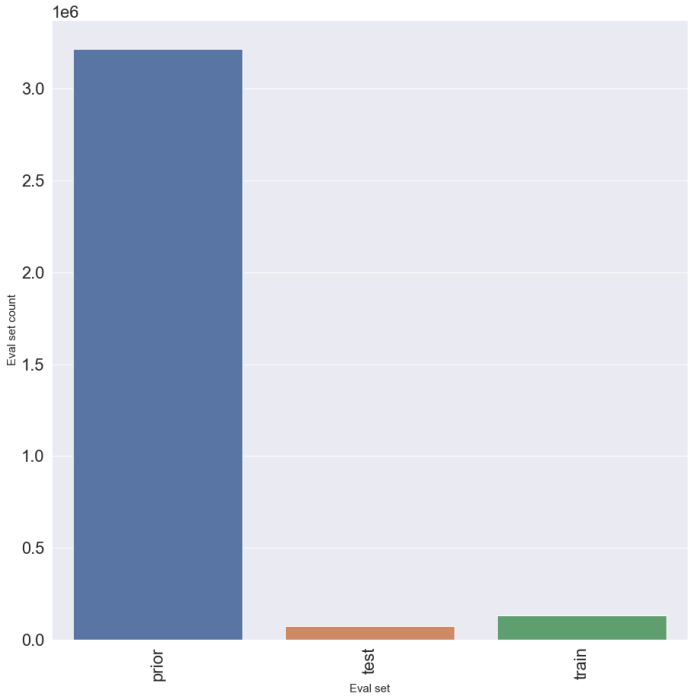
- days_since_price_order의 분포에 대해 선히스토그램으로 확인



- 결측치를 -1로 처리
- 첫 주문 이후 재주문을 할때까지 30일 이상 걸리는 회원이 제일 많음을 볼 수 있음

(2) Orders DataFrame 이상치 확인/ 처리

- eval_set column은 prior, test, train으로 총 3개의 범주로 나누어져 있음
- eval_set_column= test 는 Kaggle에서 머신러닝 test를 위해 존재하는 data임
  - 그래서, 주문 상세 정보와 관련한 order_info DataFrame에 이 데이터들이 존재하지 않음
  - 그러므로 가설 검정을 위해 eval_set=test인 데이터들을 제거함



♉ 가설 검증

1 . 주말(토,일)에 주문량이 더 많다

(1) groupby수행하여 요약변수 생성 (Total_orders)

- 요일 별 주문량을 알기 위해 'order_dow' column에 대해 groupby를 수행
  - 'Total_orders' 요일변수 생성
- 이 결과를 order_total_week DataFrame에 저장

	order_dow	Total_orders
0	0	600905
1	1	587478
2	2	467260
3	3	436972
4	4	426339
5	5	453368
6	6	448761

▶ order_total_week.head() 결과

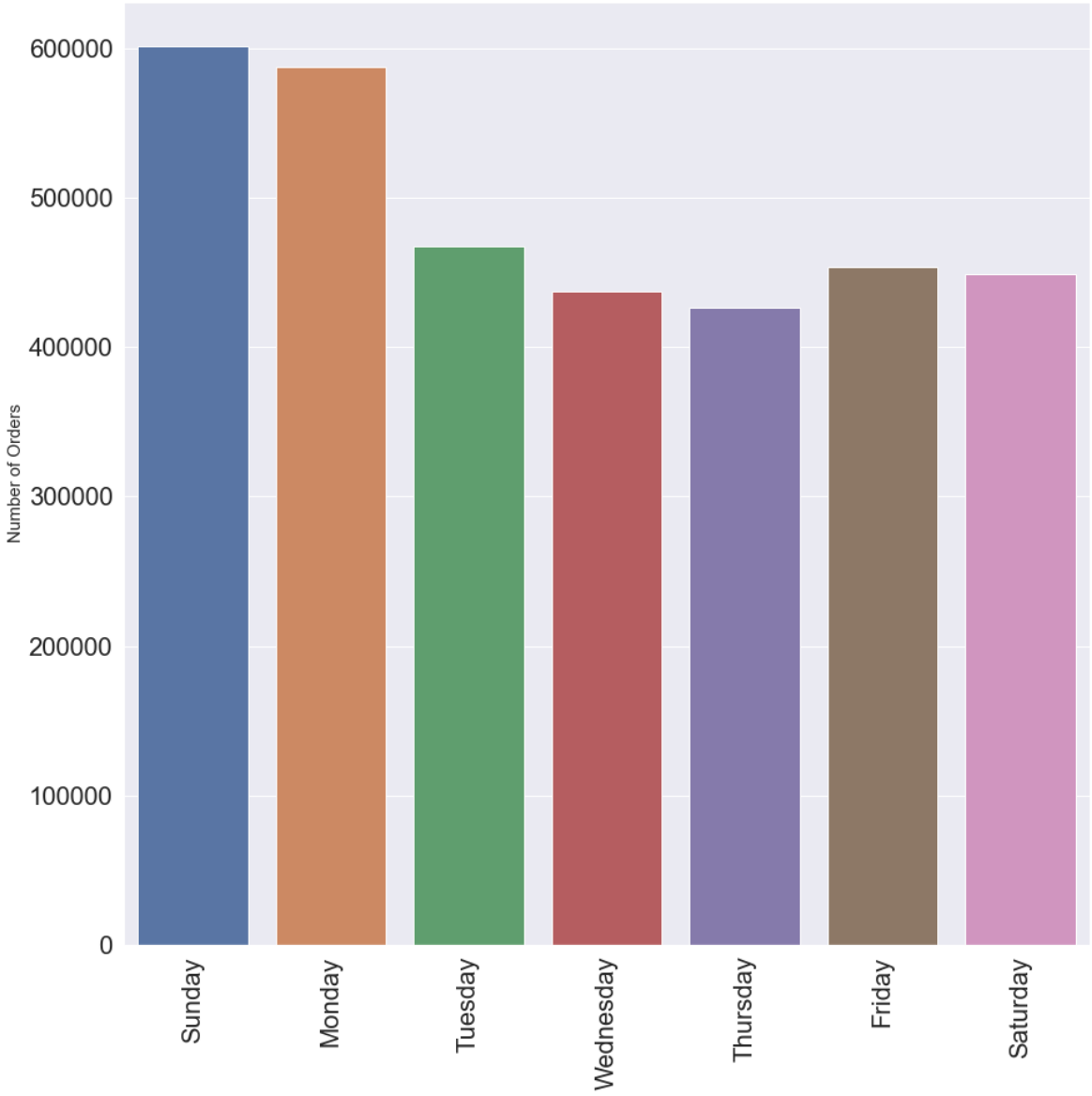
(2) order_dow를 요일 이름으로 변환하여 파생변수 생성 (day)

- order_dow=0 : Saturday ~ order_dow=6 : Sunday를 의미함
- 요일을 숫자형이 아닌 문자형으로 보기 위해서 order_dow를 요일이름으로 변환한 'day' 파생변수를 생성

	order_dow	Total_orders	day
0	0	600905	Sunday
1	1	587478	Monday
2	2	467260	Tuesday
3	3	436972	Wednesday
4	4	426339	Thursday
5	5	453368	Friday
6	6	448761	Saturday

▶ order_total_week.head() 결과

(3) 시각화



- 요일별 주문량을 히스토그램 그래프로 시각화하여 비교해 본 결과 일요일에 판매량이 600905건으로 제일 많음을 알 수 있다.

- 또한 목요일 판매량이 426339건으로 제일 적음을 알 수 있다.
- 보다 세부적으로 제품을 주문하는 때를 알기 위해 요일 & 시간별로 주문량을 살펴보자

---

2. 요일 & 시간별 주문량 알아보기

(1) 요일, 시간별 주문량을 구하기 위해 groupby 수행

- 요일, 시간 column 에 대해 groupby를 수행하고 각 그룹별 주문량을 나타내는 num_orders 요약변수 생성
- 이를 order_time DataFrame에 저장

	order_dow	order_hour_of_day	num_orders
0	0	0	3936
1	0	1	2398
2	0	2	1409
3	0	3	963
4	0	4	813

▶ order_time.head() 결과

(2) order_dow를 요일 이름으로 변환하는 파생변수 추가

- order_dow=0 : Saturday ~ order_dow=6 : Sunday를 의미함
- 요일을 숫자형이 아닌 문자형으로 보기 위해서 order_dow를 요일이름으로 변환한 'day' 파생변수를 생성

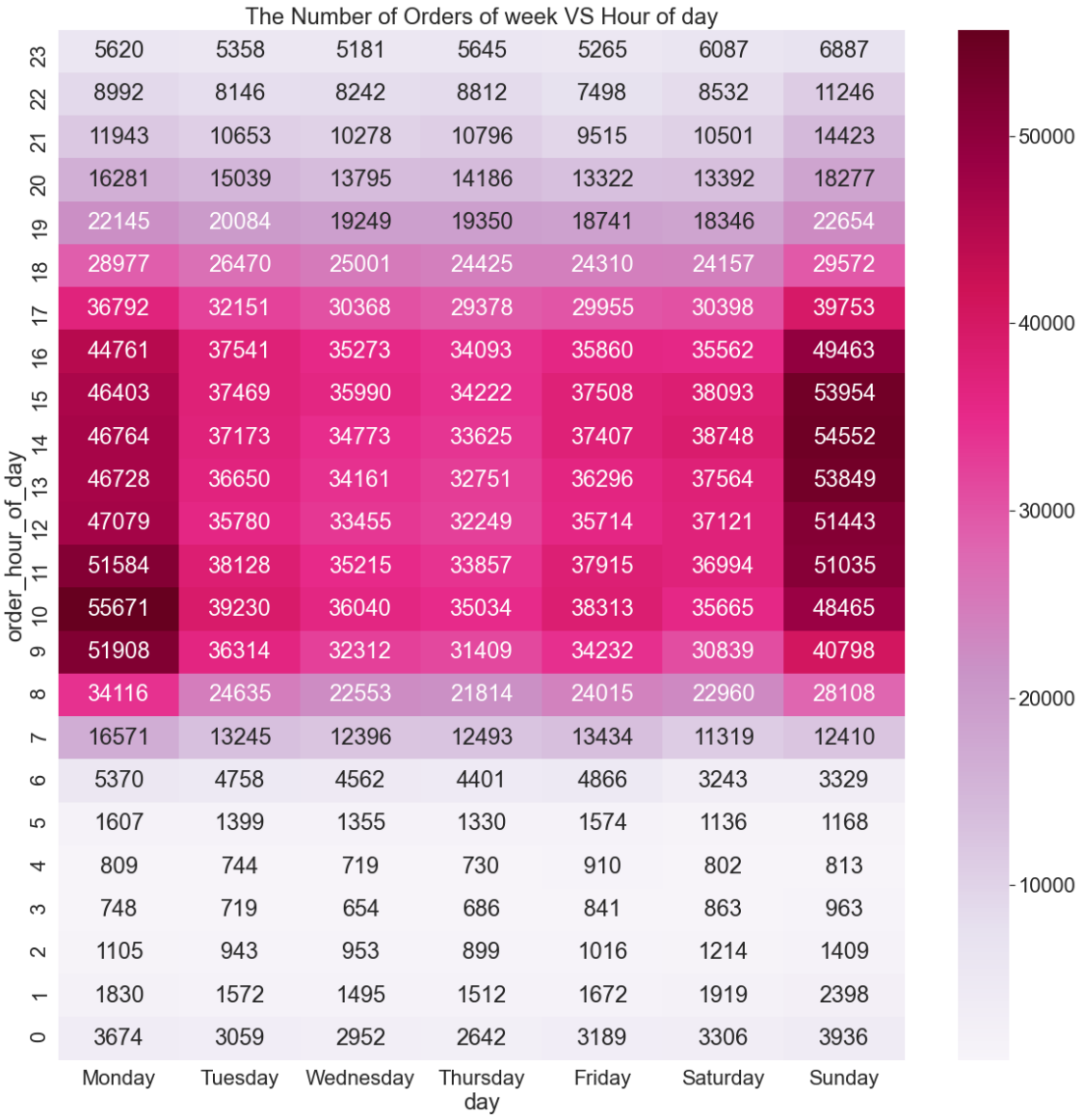
	order_dow	order_hour_of_day	num_orders	day
0	0	0	3936	Sunday
1	0	1	2398	Sunday
2	0	2	1409	Sunday
3	0	3	963	Sunday
4	0	4	813	Sunday

▶ order_time.head() 결과

(3) 요일/시간별 주문량을 pivot table로 만들어 확인

day order_hour_of_day	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
23	5620	5358	5181	5645	5265	6087	6887
22	8992	8146	8242	8812	7498	8532	11246
21	11943	10653	10278	10796	9515	10501	14423
20	16281	15039	13795	14186	13322	13392	18277
19	22145	20084	19249	19350	18741	18346	22654

(4) Visualize



- 요일 / 시간별 주문량을 Heatmap으로 표현한 결과 월요일 9시~11시와 일요일 13시~16시의 주문량이 다른 시간대에 비해 많다는 것을 알 수 있음
- 공통적으로 새벽 시간대에는 주문량이 매우 적음
- 또한, 다른 시간대에 비해 9시~15시 사이에 주문량이 많아지는 양상을 보임

3. 첫 주문 / 재주문 일 때 많이 구입한 상품은?

(1) orders DataFrame에서 첫 주문/재주문 내역 추출

- 첫 주문인 경우
  - order_number=1인 경우
  - order_number=1인 데이터를 first_order DataFrame으로 추출

	order_id	eval_set	order_number
0	2539329	prior	1
11	2168274	prior	1
26	1374495	prior	1
39	3343014	prior	1
45	2717275	prior	1

▶ first_order.head() 결과

- 재 주문인 경우
  - order_number 가 1이 아닌 경우
  - order_number !=1인 데이터를 not_first_order DataFrame으로 추출

	order_id	eval_set	order_number
1	2398795	prior	2
2	473747	prior	3
3	2254736	prior	4
4	431534	prior	5
5	3367565	prior	6

▶ not_first_order .head() 결과

(2) 첫 주문일 때 많이 구입한 상품 best 20

2-1) first_order & order_info DataFrame을 merge

- first_order & order_info DataFrame을 order_id 기준으로 merge
- 첫 주문 내역의 상세 정보 (주문한 상품: product_id)를 알기 위함
- 이를 f_order DataFrame에 저장

	order_id	product_id
0	2539329	196
1	2539329	14084
2	2539329	12427
3	2539329	26088
4	2539329	26405

▶ f_order .head() 결과

2-2) groupby 수행하여 요약변수 생성 (order_cnt)

- 첫 구매 내역에서 각 상품의 주문 횟수를 알기 위해 'product_id'를 기준으로 groupby 수행해 order_cnt 요약변수를 생성
- 이를 num_of_f_order DataFrame에 저장

	product_id	order_cnt
0	1	98
1	2	3
2	3	17
3	4	39
4	5	1

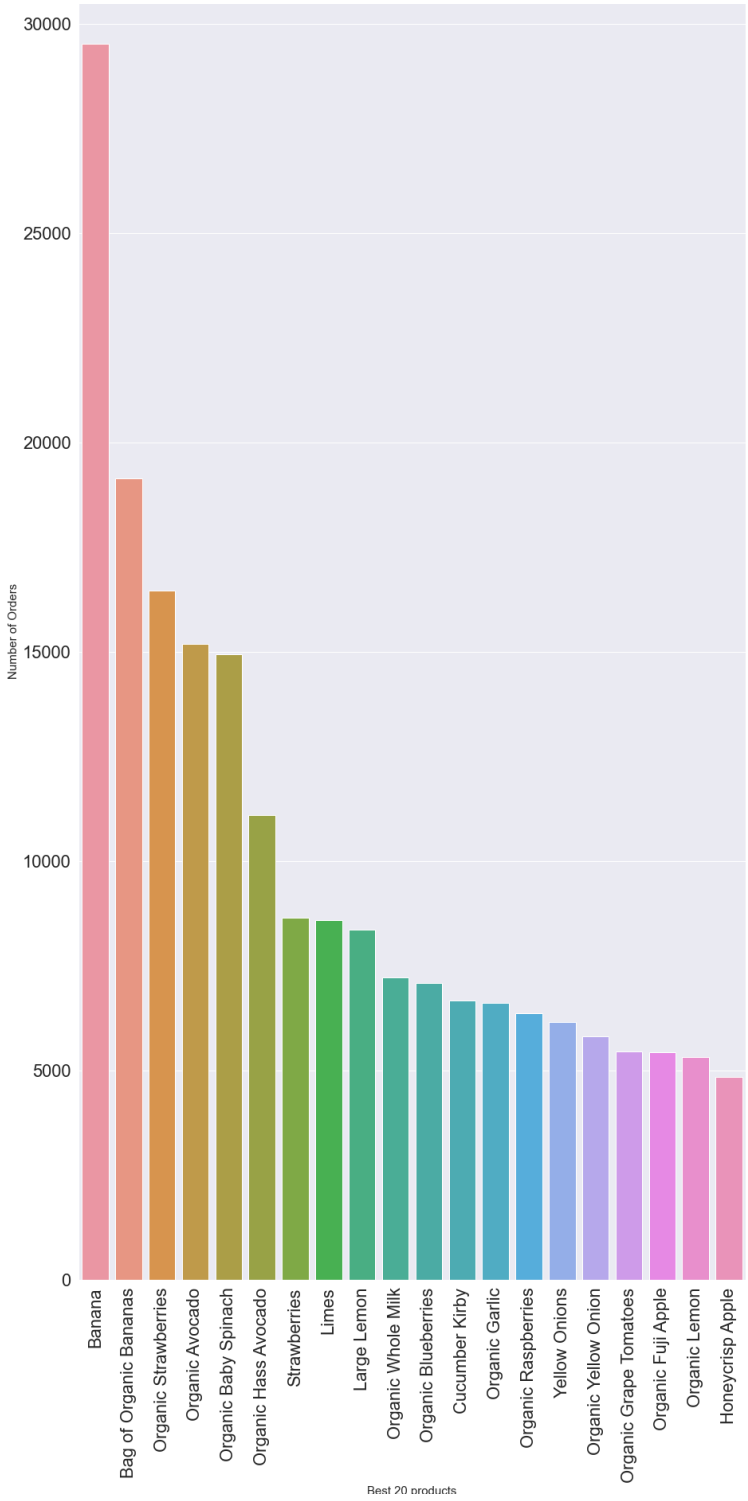
▶ num_of_f_order.head() 결과

2-3) 상품명을 알기 위해 product DataFrame과 Merge 수행

- product_id 가 가리키는 상품명을 알기위해 상품id와 상품명 data가 존재하는 product DataFrame과 merge를 수행함
- aisle_id & department_id column 제거

	product_id	order_cnt	product_name
0	24852	29534	Banana
1	13176	19158	Bag of Organic Bananas
2	21137	16464	Organic Strawberries
3	47766	15187	Organic Avocado
4	21903	14948	Organic Baby Spinach

2-4) 주문량 상위 20개의 데이터 추출 및 시각화



✔ 분석 결과

- 고객들이 처음으로 Instacart Market에서 상품을 구매할 때 바나나를 제일 많이 구매함을 알 수 있음

(3) 재 주문일 때 많이 구입한 상품 best 20

3-1) not_first_order & order_info DataFrame을 merge

- not_first_order & order_info DataFrame을 order_id 기준으로 merge
- 재 주문 내역의 상세 정보 (주문한 상품: product_id)를 알기 위함
- 이를 not_f_order DataFrame에 저장

	order_id	product_id
0	2398795	196
1	2398795	10258
2	2398795	12427
3	2398795	13176
4	2398795	26088

▶ not_f_order .head() 결과

3-2) groupby 수행하여 요약변수 생성 (order_cnt)

- 재 구매 내역에서 각 상품의 주문 횟수를 알기 위해 'product_id'를 기준으로 groupby 수행해 order_cnt 요약변수를 생성
- 이를 num_of_nf_order DataFrame에 저장

	product_id	order_cnt
0	1	1830
1	2	91
2	3	266
3	4	312
4	5	15

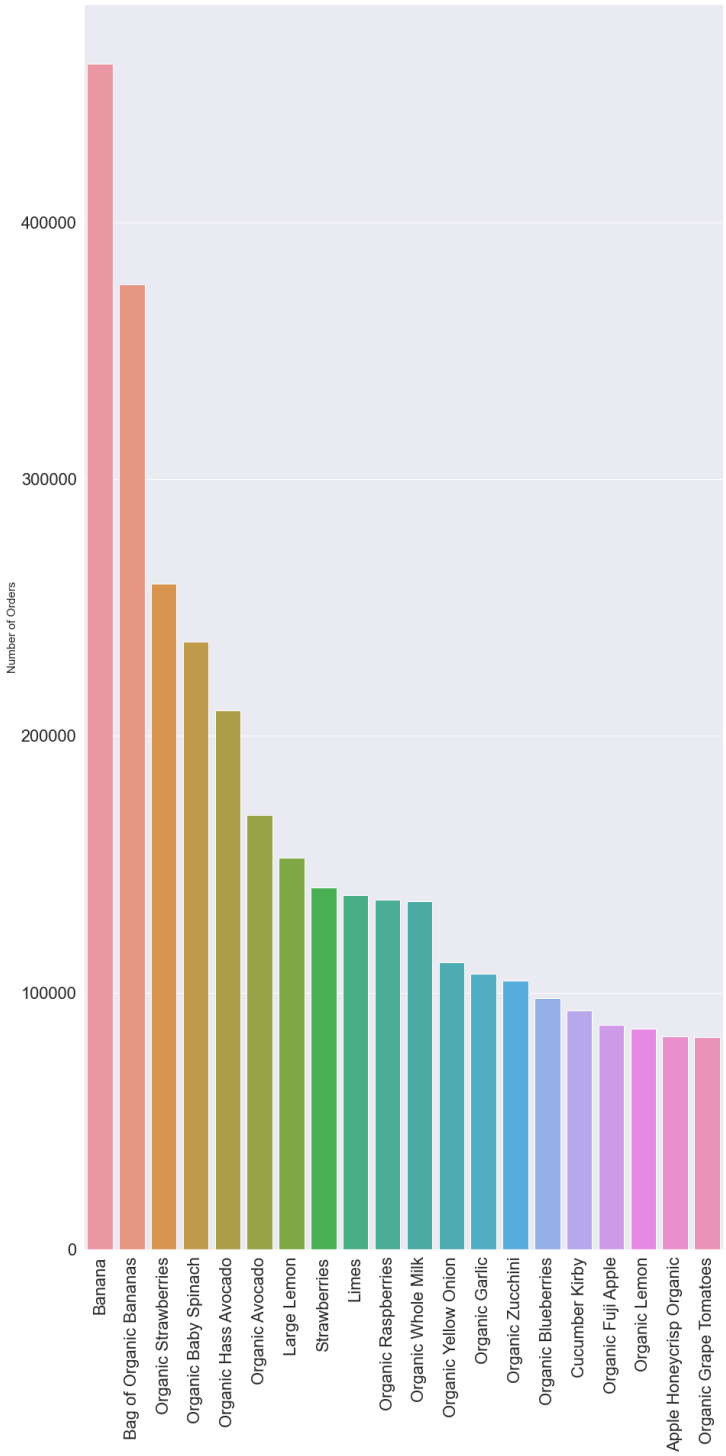
▶ num_of_nf_order .head() 결과

3-3) 상품명을 알기 위해 product DataFrame과 Merge 수행

- product_id 가 가리키는 상품명을 알기위해 상품id와 상품명 data가 존재하는 product DataFrame과 merge를 수행함
- aisle_id & department_id column 제거

	product_id	order_cnt	product_name
24841	24852	461757	Banana
13168	13176	375772	Bag of Organic Bananas
21127	21137	259113	Organic Strawberries
21893	21903	236757	Organic Baby Spinach
47180	47209	209771	Organic Hass Avocado

3-4) 주문량 상위 20개의 데이터 추출 및 시각화



✔️ 분석 결과

- 고객들이 재주문으로 Instacart Market에서 상품을 구매할 때 여전히 바나나를 제일 많이 구매함
- 첫 주문에서 많이 구매하는 상품과 재주문에서 많이 구매하는 상품에는 차이가 많이 있지 않음

---

4. 첫주문일때 주문 상품의 갯수보다 재주문일때 주문 상품의 갯수에 차이가 없다

(1) 첫주문 내역에서 1개의 주문 내역 당 판매하는 물건의 갯수

1-1) 데이터 준비

- 첫 주문일 때의 주문 내역
- f_order DataFrame을 이용함

	order_id	product_id
0	2539329	196
1	2539329	14084
2	2539329	12427
3	2539329	26088
4	2539329	26405

▶ f_order .head() 결과

1-2) groupby 를 수행하여 요약변수 생성 (product_number)

- 각각의 주문내역에 대해 주문한 상품의 갯수를 구하기 위해 'order_id'를 기준으로 groupby를 수행해 요약변수 product_number 을 생성함

	order_id	product_number
0	20	8
1	35	5
2	37	3
3	57	7
4	75	16

(2) 재주문 내역에서 1개의 주문 내역 당 판매하는 물건의 갯수

2-1) 데이터 준비

- 재 주문일 때의 주문 내역
- nf_order DataFrame을 이용함

	order_id	product_id
0	2398795	196
1	2398795	10258
2	2398795	12427
3	2398795	13176
4	2398795	26088

▶ nf_order.head() 결과

2-2) groupby 를 수행하여 요약변수 생성 (product_number)

- 각각의 주문내역에 대해 주문한 상품의 갯수를 구하기 위해 'order_id'를 기준으로 groupby를 수행해 요약변수 product_number 을 생성함

	order_id	product_number
0	1	8
1	2	9
2	3	8
3	4	13
4	5	26

(3) 첫주문 내역과 재주문 내역에서 판매량에 대한 등분산 검정 실시

```
LeveneResult(statistic=0.3494983766953213, pvalue=0.5543972642090749)
```

✔️ 분석 결과

- 등분산 검정 결과 F값이 0.349이고 p-value=0.554 이므로 유의수준 0.05에서 귀무가설을 기각하지 않음
- 그러므로 두 표본(첫 판매량, 이후 판매량)의 분산이 동일
- 등분산이므로 equal_var=True로 입력하고 독립표본 t-검정을 진행

(4) 독립표본 t검정 결과

```
Ttest_indResult(statistic=-1.8390251739443826, pvalue=0.06591157202764523)
```

✔️ 분석 결과

- 독립표본 t-검정 결과 t값이 -1.839 , p-value=0.065이므로 유의수준 0.05에서 "첫주문과 재주문 에 따른 평균 총 주문량에는 차이가 없다"는 가설을 기각하지 않음
- 따라서 첫주문과 재주문에 따른 평균 주문량에는 차이가 없음