Shopify Summer 2022 Data Science Intern Challenge

Question 1

EDA

In [2]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore') # just to make the output nicer without IPython warnings w
hen creating a new column
```

Firstly, let's perform an EDA to get a better understanding of the distribution and characteristics of the dataset

In [3]:
```
data = pd.read_csv('data.csv')
data.shape # check number of observations and features
```

Out[3]:
```
(5000, 7)
```

In [4]:
```
data.head() # a glimpse of the dataset
```

Out[4]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
0	1	53	746	224	2	cash	2017-03-13 12:36:56
1	2	92	925	90	1	cash	2017-03-03 17:38:52
2	3	44	861	144	1	cash	2017-03-14 4:23:56
3	4	18	935	156	1	credit_card	2017-03-26 12:43:37
4	5	18	883	156	1	credit_card	2017-03-01 4:35:11

In [5]:
```
data.isna().sum() # checking missing values
```

Out[5]:
```
order_id          0
shop_id           0
user_id           0
order_amount      0
total_items       0
payment_method    0
created_at        0
dtype: int64
```

In [6]:
```
data[['order_amount', 'total_items']].describe() # overview of statistical metrics for the m
ost important two features
```

Out[6]:

	order_amount	total_items
count	5000.000000	5000.00000
mean	3145.128000	8.78720
std	41282.539349	116.32032
min	90.000000	1.00000
25%	163.000000	1.00000
50%	284.000000	2.00000
75%	390.000000	3.00000
max	704000.000000	2000.00000

From above we see that indeed the AOV is $3145.13, but the max order amount ($704000) and max total items (2000) are both extremely large, causing unusually large standard deviations. The cause for such observations will be investigated in more detail later, but now we already see that there are outliers in the dataset which influence the average significantly, our AOV is driven up by such extremely large order amounts, and thus is not a proper metric to reflect the true characteristics of the dataset

a) Reasons for nonsensical AOV and a better way to evaluate the dataset

Reason 1: Shop 78 with overpriced sneakers

Reason 2: Shop 42 with bulk sale

As we've seen in the statistical description, we have problematic observations that deviate severely from the rest of the dataset. Let's find them now, starting with checking the unit price for sneakers in each shop (according to the question, each shop only sells one model, so it has only one price)

In [7]:
```
data_copy = data[['shop_id', 'order_amount', 'total_items']] # let's not change the original
dataset
data_copy['unit_price'] = data_copy.order_amount/data_copy.total_items # find unit price for
each shop
data_copy = data_copy.groupby(['shop_id']).mean().sort_values(by = ['unit_price'], ascending
= False) # group by shop then sort unit price
data_copy.head()
```

Out[7]:

shop_id	order_amount	total_items	unit_price
78	49213.043478	1.913043	25725.0
42	235101.490196	667.901961	352.0
12	352.698113	1.754717	201.0
89	379.147541	1.934426	196.0
99	339.444444	1.740741	195.0

In [8]:
```
data_copy.unit_price.describe()
```

Out[8]:
```
count      100.000000
mean       407.990000
std       2557.462906
min         90.000000
25%        132.750000
50%        153.000000
75%        168.250000
max      25725.000000
Name: unit_price, dtype: float64
```

We see that almost every shop sells sneakers at a reasonable price, ranging from $90 to $352, but shop 78 sells sneakers at the price of $25725 (luxurious shop I guess, or money laundry), which significantly drives up the AOV.

But shop 78 isn't the only cause of the problem. If we take a closer look at the total_items column in the previous table, we see shop 42 sells 668 pairs of sneakers on average per order while other shops sell around 2 on average. Let's dig into it

In [9]:
```
data_copy.sort_values(by=['total_items'], ascending=False).head()
```

Out[9]:

shop_id	order_amount	total_items	unit_price
42	235101.490196	667.901961	352.0
37	340.208333	2.395833	142.0
24	320.727273	2.290909	140.0
90	403.224490	2.265306	178.0
10	332.301887	2.245283	148.0

In [10]:
```
data_copy.total_items.describe()
```

Out[10]:
```
count    100.000000
mean       8.652863
std       66.590946
min        1.731707
25%        1.912724
50%        1.981125
75%        2.076250
max      667.901961
Name: total_items, dtype: float64
```

We see that almost every shop sells 2 sneakers on average, but shop 42 sells 667 pairs of sneakers on average within the past 30 days, which, again, significantly deviates from the rest. Now if we take an even closer look at the detailed transction information of shop 42 (I simply skimmed through the dataset by sorting the total_items column), we immediately spot repetitive large transctions (2000 pairs of sneakers) from user 607

In [11]:
```
data.loc[data.user_id  == 607].head()
```

Out[11]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
15	16	42	607	704000	2000	credit_card	2017-03-07 4:00:00
60	61	42	607	704000	2000	credit_card	2017-03-04 4:00:00
520	521	42	607	704000	2000	credit_card	2017-03-02 4:00:00
1104	1105	42	607	704000	2000	credit_card	2017-03-24 4:00:00
1362	1363	42	607	704000	2000	credit_card	2017-03-15 4:00:00

In [12]:
```
data.loc[(data.shop_id == 42) & (data.user_id != 607)].head()
```

Out[12]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
40	41	42	793	352	1	credit_card	2017-03-24 14:15:41
308	309	42	770	352	1	credit_card	2017-03-11 18:14:39
409	410	42	904	704	2	credit_card	2017-03-04 14:32:58
834	835	42	792	352	1	cash	2017-03-25 21:31:25
835	836	42	819	704	2	cash	2017-03-09 14:15:15

We see that one user with id 607 buys 2000 sneakers at a fixed time (4:00:00) for many times, a reasonable explanation for this phenomenon is that this is not the behavior of normal customer who walks in the shop or buys some sneakers for personal wear or gifting, but an automatic transction from other suppliers (or shops) who need large quantity for second-hand selling. Notice that there are actually stll orders of one or two pairs of sneakers from normal customers, which is why we see the average total_items in shop 42 is not 2000 but 668

In summary, the naively calculated AOV doesn't take into account the case of shop 78 which sells extremely overpriced sneakers and shop 42 which offers bulk sale at the quantity of 2000. Both shops significantly increase the order_amount, which consequently drives up the AOV to a value that doesn't make sense.

A better way to evaluate this data:

If we still want to analyze AOV and our focus is the pattern of normal customers who buy regularly priced sneakers, we'd better not include observations from shop 78 and bulk sale transctions from shop 42. By removing such observations which significantly deviate from the rest, we can better focus on our goal of understanding the purchase habit of regular customers.

But if we want to fully understand this dataset, we can't simply ignore the fact that there are overpriced sneakers and people who are willing to pay for them, and there are bulk sales happening. Although they don't fit into the normal pattern, they are still valid observations and they carry valuable information. Depending on the focus of the analysis, we may need to consider them in our calculation as well.

b) What metric would I report

Although I believe we should evaluate as many metrics as possible for a better understanding of the dataset, if I'm asked to report only one metric, I would report **median**, which is 0.5 quantile of the order_amount. It's not affected by the extremely large order_amount since it represents the middle number of all 5000 observations, several hundreds of extreme values at the tail won't influence the middle number. A quick comparison is below to show that the existence of extreme observations from shop 42 and shop 78 won't influence the median

In [13]:
```
median_w = np.median(data.order_amount)
median_wo = np.median(data[((data.shop_id != 42) & data.user_id!= 607) & (data.shop_id!=78)]
.order_amount) # remove shop 42 user 607 data and shop 78 data
median_w == median_wo
```

Out[13]:
```
True
```

c) What's the value

In [14]:
```
median_w
```

Out[14]:
```
284.0
```

The median is $284, which is a number that makes sense for sneakers. It's the price for either several pairs of cheap sneakers (like $90) or one pair of more expensive sneakers

Question 2

a) How many orders were shipped by Speedy Express in total

In [ ]:
```
SELECT COUNT(ShipperName)
FROM Orders o
LEFT JOIN Shippers s
ON s.ShipperID = o.ShipperID
Where ShipperName = 'Speedy Express'
```

Answer: 54

b) What is the last name of the employee with the most orders

In [ ]:
```
WITH NumOrder AS (
SELECT EmployeeID, Count(EmployeeID) Num
FROM orders
GROUP BY EmployeeID
ORDER BY Num DESC
)
SELECT LastName
FROM NumOrder no
LEFT JOIN Employees e
ON no.EmployeeID = e.EmployeeID
LIMIT 1
```

Answer: Peacock

c) What product was ordered the most by customers in Germany

In [ ]:
```
WITH OrderGermany AS (
SELECT od.ProductID, SUM(od.Quantity) Num
FROM Orders o
LEFT JOIN Customers c ON o.CustomerID = c.CustomerID
LEFT JOIN OrderDetails od ON o.OrderID = od.OrderID
WHERE c.Country = 'Germany'
GROUP BY od.ProductID
ORDER BY Num DESC
)
SELECT p.ProductName
FROM OrderGermany og
LEFT JOIN Products p
ON p.ProductID = og.ProductID
LIMIT 1
```

Answer: Boston Crab Meat