shopify

January 19, 2022

1 Summer 2022 Data Science Intern Challenge

1.1 Question 1

On Shopify, we have exactly 100 sneaker shops, and each of these shops sells only one model of shoe. We want to do some analysis of the average order value (AOV). When we look at orders data over a 30 day window, we naively calculate an AOV of $3145.13. Given that we know these shops are selling sneakers, a relatively affordable item, something seems wrong with our analysis.

a. Think about what could be going wrong with our calculation. Think about a better way to evaluate this data.

[47]:
```
# Import packages
import pandas as pd
import matplotlib.pyplot as plt
```

[48]:
```
# Import csv file
sneakers = pd.read_csv('2019 Winter Data Science Intern Challenge Data Set -␣
↪Sheet1.csv')
```

An AOV of $3145.13 seems very unreasonble for a sneaker store. Before investigating what goes wrong, I would like to check the data quality first. There are no missing values or duplicating order ids, so we are good to proceed.

[49]:
```
sneakers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   order_id        5000 non-null   int64
 1   shop_id         5000 non-null   int64
 2   user_id         5000 non-null   int64
 3   order_amount    5000 non-null   int64
 4   total_items     5000 non-null   int64
 5   payment_method  5000 non-null   object
 6   created_at      5000 non-null   object
dtypes: int64(5), object(2)
memory usage: 273.6+ KB
```

[50]:
```
sneakers[sneakers.duplicated()]
```

[50]:
```
Empty DataFrame
Columns: [order_id, shop_id, user_id, order_amount, total_items, payment_method,
created_at]
Index: []
```

[51]:
```
sneakers.describe()
```

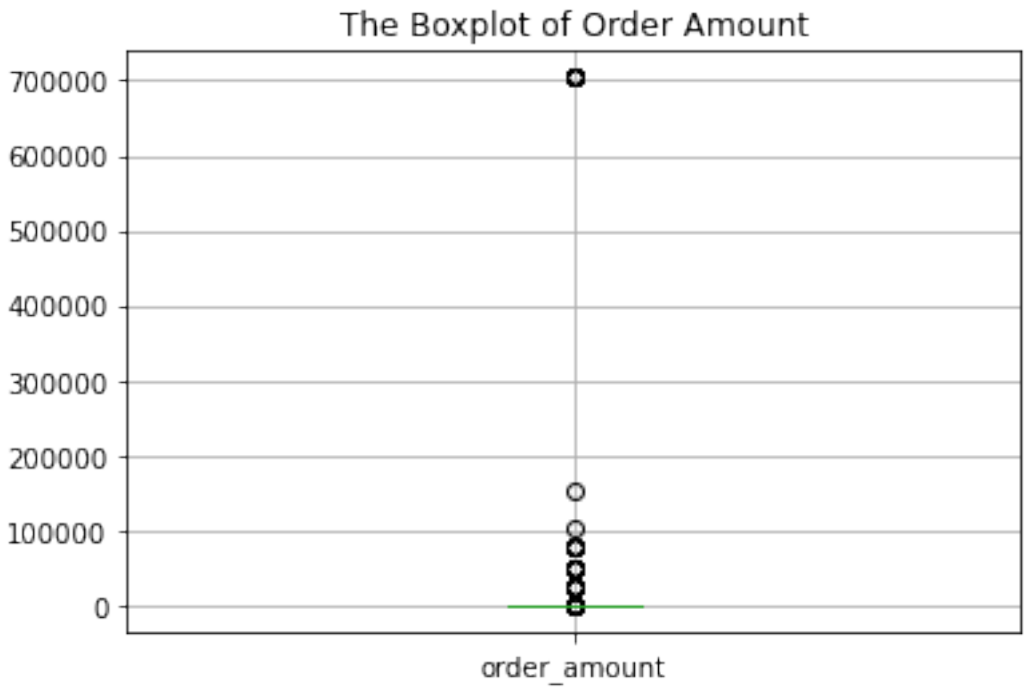
[51]:

	order_id	shop_id	user_id	order_amount	total_items
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720
std	1443.520003	29.006118	87.798982	41282.539349	116.32032
min	1.000000	1.000000	607.000000	90.000000	1.00000
25%	1250.750000	24.000000	775.000000	163.000000	1.00000
50%	2500.500000	50.000000	849.000000	284.000000	2.00000
75%	3750.250000	75.000000	925.000000	390.000000	3.00000
max	5000.000000	100.000000	999.000000	704000.000000	2000.00000

I think the abnormal AOV is due to the outliers in the order amounts. By running the `describe()` function from pandas, we can tell the AOV is indeed \$3145.13. Even the 75% percentile is \$390, which looks normal for a sneakers store. However, I notice that the maximum order amount is \$704,000, which is much higher than the other values and looks unusual. The standard deviation for order amount is 41282.54, indicating that the data is widely spread. These numbers could be considered outliers in the data.

Next, I want to see the distribution of the data.

[52]:
```
sneakers.boxplot(column='order_amount')
plt.title('The Boxplot of Order Amount')
plt.show()
```



The boxplot looks extremely right-skewed, and the box can be barely seen. I will find out these amounts by grouping the data by order amount and counting the number of occurances for each group.

[53]:
```
unique_amounts = sneakers.groupby(['order_amount']).size().
 ↪reset_index(name='count').\
sort_values(by='order_amount', ascending=False)
unique_amounts.head(10)
```

[53]:

	order_amount	count
257	704000	17
256	154350	1
255	102900	1
254	77175	9
253	51450	16
252	25725	19
251	1760	1
250	1408	2
249	1086	1
248	1064	1

First, we can tell that some amounts appear multiple times, such as 704000, 77175, 51450 and 25725. We want to take a closer look at these number. I will filter out the entries with these amounts for further investigation. The next highest order amount 1760 seems reasonable for a sneakers store, and we might keep the data after it.

[55]:
```
pd.set_option('display.max_rows', 61)
sneakers.loc[sneakers['order_amount'].isin([704000, 77175, 51450, 25725])].\
sort_values(by=['order_amount', 'created_at'], ascending=False)
```

[55]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method
2835	2836	42	607	704000	2000	credit_card
2969	2970	42	607	704000	2000	credit_card
4056	4057	42	607	704000	2000	credit_card
4882	4883	42	607	704000	2000	credit_card
1104	1105	42	607	704000	2000	credit_card
3332	3333	42	607	704000	2000	credit_card
4868	4869	42	607	704000	2000	credit_card
1562	1563	42	607	704000	2000	credit_card
1602	1603	42	607	704000	2000	credit_card
1362	1363	42	607	704000	2000	credit_card
2153	2154	42	607	704000	2000	credit_card
1436	1437	42	607	704000	2000	credit_card
15	16	42	607	704000	2000	credit_card
2297	2298	42	607	704000	2000	credit_card
60	61	42	607	704000	2000	credit_card
520	521	42	607	704000	2000	credit_card
4646	4647	42	607	704000	2000	credit_card
1259	1260	78	775	77175	3	credit_card
2564	2565	78	915	77175	3	debit
2690	2691	78	962	77175	3	debit
4192	4193	78	787	77175	3	credit_card
3403	3404	78	928	77175	3	debit
2906	2907	78	817	77175	3	debit
3724	3725	78	766	77175	3	credit_card
4420	4421	78	969	77175	3	debit
4715	4716	78	818	77175	3	debit
1529	1530	78	810	51450	2	cash
2452	2453	78	709	51450	2	cash
2495	2496	78	707	51450	2	cash
490	491	78	936	51450	2	debit
3101	3102	78	855	51450	2	credit_card
4079	4080	78	946	51450	2	cash
2512	2513	78	935	51450	2	debit
617	618	78	760	51450	2	cash
2818	2819	78	869	51450	2	debit
493	494	78	983	51450	2	cash
3705	3706	78	828	51450	2	credit_card
3167	3168	78	927	51450	2	cash
511	512	78	967	51450	2	cash
4412	4413	78	756	51450	2	debit
2821	2822	78	814	51450	2	cash
4311	4312	78	960	51450	2	debit

1419	1420	78	912	25725	1	cash
3085	3086	78	910	25725	1	cash
2773	2774	78	890	25725	1	cash
4584	4585	78	997	25725	1	cash
4505	4506	78	866	25725	1	debit
3440	3441	78	982	25725	1	debit
3151	3152	78	745	25725	1	credit_card
1204	1205	78	970	25725	1	credit_card
2548	2549	78	861	25725	1	cash
1452	1453	78	812	25725	1	credit_card
1384	1385	78	867	25725	1	cash
1193	1194	78	944	25725	1	debit
4918	4919	78	823	25725	1	cash
1056	1057	78	800	25725	1	debit
2270	2271	78	855	25725	1	credit_card
160	161	78	990	25725	1	credit_card
2922	2923	78	740	25725	1	debit
3780	3781	78	889	25725	1	cash
4040	4041	78	852	25725	1	cash

	created_at
2835	2017-03-28 4:00:00
2969	2017-03-28 4:00:00
4056	2017-03-28 4:00:00
4882	2017-03-25 4:00:00
1104	2017-03-24 4:00:00
3332	2017-03-24 4:00:00
4868	2017-03-22 4:00:00
1562	2017-03-19 4:00:00
1602	2017-03-17 4:00:00
1362	2017-03-15 4:00:00
2153	2017-03-12 4:00:00
1436	2017-03-11 4:00:00
15	2017-03-07 4:00:00
2297	2017-03-07 4:00:00
60	2017-03-04 4:00:00
520	2017-03-02 4:00:00
4646	2017-03-02 4:00:00
1259	2017-03-27 9:27:20
2564	2017-03-25 1:19:35
2690	2017-03-22 7:33:25
4192	2017-03-18 9:25:32
3403	2017-03-16 9:45:05
2906	2017-03-16 3:45:46
3724	2017-03-16 14:13:26
4420	2017-03-09 15:21:35
4715	2017-03-05 5:10:44

```
1529   2017-03-29 7:12:01
2452  2017-03-27 11:04:04
2495   2017-03-26 4:38:52
490   2017-03-26 17:08:19
3101   2017-03-21 5:10:34
4079  2017-03-20 21:14:00
2512  2017-03-18 18:57:13
617   2017-03-18 11:18:42
2818   2017-03-17 6:25:51
493   2017-03-16 21:39:35
3705  2017-03-14 20:43:15
3167  2017-03-12 12:23:08
511    2017-03-09 7:23:14
4412   2017-03-02 4:13:39
2821  2017-03-02 17:13:25
4311   2017-03-01 3:02:10
1419  2017-03-30 12:23:43
3085   2017-03-26 1:59:27
2773  2017-03-26 10:36:43
4584  2017-03-25 21:48:44
4505  2017-03-22 22:06:01
3440  2017-03-19 19:02:54
3151  2017-03-18 13:13:07
1204  2017-03-17 22:32:21
2548  2017-03-17 19:36:00
1452  2017-03-17 18:09:54
1384  2017-03-17 16:38:06
1193  2017-03-16 16:38:26
4918  2017-03-15 13:26:46
1056  2017-03-15 10:16:45
2270  2017-03-14 23:58:22
160    2017-03-12 5:56:57
2922  2017-03-12 20:10:58
3780  2017-03-11 21:14:50
4040  2017-03-02 14:31:12
```

We first notice that the order amount of 704000 is purchased by the same user with user id 607, and the order is made at 4am every day with a credit card. This could be a potential fraud, or the result of error in data management. Also, the order amounts of 77175, 51450 and 25725 are all from the same shop, with shop id 78. They are only different in item quantity. It's interesting to find out all the orders for shop 78.

[68]:
```
sneakers.loc[sneakers['shop_id'].isin([78])].sort_values(by=['created_at'],␣
↪ascending=True)
```

[68]:
```
      order_id  shop_id  user_id  order_amount  total_items payment_method  \
4311      4312       78      960         51450            2          debit
```

4040	4041	78	852	25725	1	cash
2821	2822	78	814	51450	2	cash
4412	4413	78	756	51450	2	debit
2492	2493	78	834	102900	4	debit
4715	4716	78	818	77175	3	debit
4420	4421	78	969	77175	3	debit
511	512	78	967	51450	2	cash
3780	3781	78	889	25725	1	cash
3167	3168	78	927	51450	2	cash
2922	2923	78	740	25725	1	debit
160	161	78	990	25725	1	credit_card
3705	3706	78	828	51450	2	credit_card
2270	2271	78	855	25725	1	credit_card
1056	1057	78	800	25725	1	debit
4918	4919	78	823	25725	1	cash
3724	3725	78	766	77175	3	credit_card
1193	1194	78	944	25725	1	debit
493	494	78	983	51450	2	cash
2906	2907	78	817	77175	3	debit
3403	3404	78	928	77175	3	debit
1384	1385	78	867	25725	1	cash
1452	1453	78	812	25725	1	credit_card
2548	2549	78	861	25725	1	cash
1204	1205	78	970	25725	1	credit_card
2818	2819	78	869	51450	2	debit
617	618	78	760	51450	2	cash
3151	3152	78	745	25725	1	credit_card
2512	2513	78	935	51450	2	debit
4192	4193	78	787	77175	3	credit_card
3440	3441	78	982	25725	1	debit
4079	4080	78	946	51450	2	cash
3101	3102	78	855	51450	2	credit_card
4505	4506	78	866	25725	1	debit
2690	2691	78	962	77175	3	debit
2564	2565	78	915	77175	3	debit
4584	4585	78	997	25725	1	cash
2773	2774	78	890	25725	1	cash
490	491	78	936	51450	2	debit
3085	3086	78	910	25725	1	cash
2495	2496	78	707	51450	2	cash
2452	2453	78	709	51450	2	cash
691	692	78	878	154350	6	debit
1259	1260	78	775	77175	3	credit_card
1529	1530	78	810	51450	2	cash
1419	1420	78	912	25725	1	cash

created_at

```
4311   2017-03-01 3:02:10
4040  2017-03-02 14:31:12
2821  2017-03-02 17:13:25
4412   2017-03-02 4:13:39
2492   2017-03-04 4:37:34
4715   2017-03-05 5:10:44
4420  2017-03-09 15:21:35
511    2017-03-09 7:23:14
3780  2017-03-11 21:14:50
3167  2017-03-12 12:23:08
2922  2017-03-12 20:10:58
160    2017-03-12 5:56:57
3705  2017-03-14 20:43:15
2270  2017-03-14 23:58:22
1056  2017-03-15 10:16:45
4918  2017-03-15 13:26:46
3724  2017-03-16 14:13:26
1193  2017-03-16 16:38:26
493   2017-03-16 21:39:35
2906   2017-03-16 3:45:46
3403   2017-03-16 9:45:05
1384  2017-03-17 16:38:06
1452  2017-03-17 18:09:54
2548  2017-03-17 19:36:00
1204  2017-03-17 22:32:21
2818   2017-03-17 6:25:51
617   2017-03-18 11:18:42
3151  2017-03-18 13:13:07
2512  2017-03-18 18:57:13
4192   2017-03-18 9:25:32
3440  2017-03-19 19:02:54
4079  2017-03-20 21:14:00
3101   2017-03-21 5:10:34
4505  2017-03-22 22:06:01
2690   2017-03-22 7:33:25
2564   2017-03-25 1:19:35
4584  2017-03-25 21:48:44
2773  2017-03-26 10:36:43
490   2017-03-26 17:08:19
3085   2017-03-26 1:59:27
2495   2017-03-26 4:38:52
2452  2017-03-27 11:04:04
691   2017-03-27 22:51:43
1259   2017-03-27 9:27:20
1529   2017-03-29 7:12:01
1419  2017-03-30 12:23:43
```

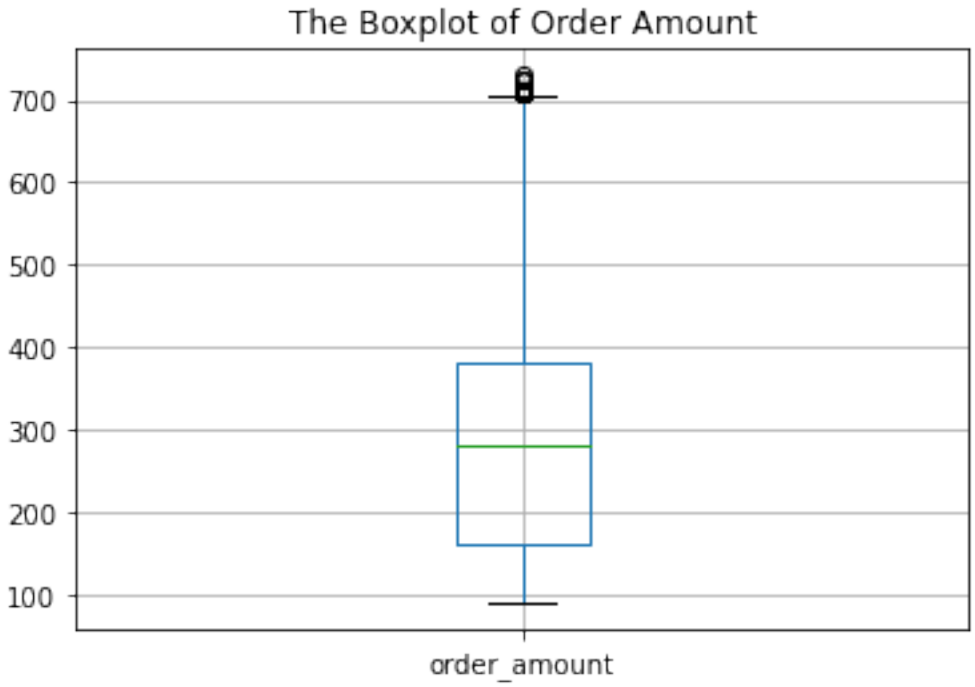
The order amounts of this shop are all very high, and the unit price of the item it's selling is $25725, which is a lot higher than regular sneakers. Thus, this shop could be flagged as fraud as well.

To make sure AOV reflects the dataset well enough, we could remove the outliers. I decide to remove all the rows with order amount that is outside the 1.5 times interquartile range.

[69]:
```
def remove_outliers(df):
    first = df.order_amount.quantile(0.25)
    third = df.order_amount.quantile(0.75)
    iqr = third - first
    return df[(df.order_amount >= first - 1.5 * iqr) & (df.order_amount <=␣
 ↪third + 1.5 * iqr)]

sneakers_clean = remove_outliers(sneakers)
```

[70]:
```
sneakers_clean.boxplot(column='order_amount')
plt.title('The Boxplot of Order Amount')
plt.show()
```



[62]:
```
sneakers_clean.describe()
```

[62]:

	order_id	shop_id	user_id	order_amount	total_items
count	4859.000000	4859.000000	4859.000000	4859.000000	4859.000000
mean	2497.395966	49.852645	849.905742	293.715374	1.950196
std	1443.356555	29.049171	86.887496	144.453395	0.919791

```
min       1.000000     1.000000   700.000000     90.000000     1.000000
25%    1244.500000    24.000000   776.000000    162.000000     1.000000
50%    2498.000000    50.000000   850.000000    280.000000     2.000000
75%    3749.500000    74.000000   925.000000    380.000000     3.000000
max    5000.000000   100.000000   999.000000    730.000000     5.000000
```

After removing the outliers, the boxplot represents the dataset better, but it's still right-skewed, with a tail on the right. By running the `describe()` function again, we can see the new AOV is $293.72 with a standard deviation of 144.46.

b. What metric would you report for this dataset?

Since the distribution of the order amount is still right-skewed, I think it would be better to use the Median Order Value (MOV) as the metric instead of AOV. A median is better to eliminate outliers on the high or low end of the data.

c. What is its value?

From the results of the `describe()` function, the MOV is $280.

1.2 Question 2

For this question you'll need to use SQL. Follow this link to access the data set required for the challenge. Please use queries to answer the following questions. Paste your queries along with your final numerical answers below.

a. How many orders were shipped by Speedy Express in total?

```
SELECT COUNT(DISTINCT OrderID) FROM
Orders A LEFT JOIN Shippers B
ON A.ShipperID = B.ShipperID
WHERE ShipperName = 'Speedy Express';
```

54 orders were shipped by Speedy Express in total.

b. What is the last name of the employee with the most orders?

```
SELECT LastName, Max(Number) FROM
(SELECT *, COUNT(EmployeeID) AS Number FROM
(SELECT A.OrderID, A.EmployeeID, B.LastName FROM
Orders A LEFT JOIN Employees B
ON A.EmployeeID = B.EmployeeID)
GROUP BY EmployeeID);
```

The employee with last name Peacock has the most order, and the number of orders is 40.

c. What product was ordered the most by customers in Germany?

```
SELECT ProductName, MaxNumOrders FROM
(SELECT ProductID, MAX(NumOrders) AS MaxNumOrders FROM
(SELECT ProductID, SUM(Quantity) AS NumOrders FROM
(SELECT * FROM
Orders A INNER JOIN OrderDetails B
ON A.OrderID = B.OrderID
```

```
WHERE CustomerID IN
(SELECT CustomerID FROM
Customers WHERE Country = 'Germany'))
GROUP BY ProductID)) C
INNER JOIN Products D
ON C.ProductID = D.ProductID;
```

Boston Crab Meat was ordered the most by customers in Germany, and 160 units were sold.