Analysis of Sneaker Data - Shopify

- In this notebook I have performed required analysis the answer the questions.
- Please find the answer to the questions below. Supporting analysis can be found further down in the notebook

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

- The first though that comes to mind when seeing a extremely high average order value is that there must be some orders with huge amounts which is causing this to happen. When AOV is calculated naively then it does not account for the outliers in the data which seem to be highly affecting the AOV.

(b) What metric would you report for this dataset?

- After all the analysis, it can be seen that the median value provides a good measure for the AOV and it is robust to outliers.

(c) What is its value?

- The AOV based on the median value is $284.0 for the given orders.

```
import numpy as np
import pandas as pd
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
data = pd.read_csv("drive/My Drive/shopify/2019 Winter Data Science Intern Challenge Data Set
data.head()
```

	order_id	shop_id	user_id	order_amount	total_items	payment_method	createc
0	1	53	746	224	2	cash	2017-03-13 12:3
1	2	92	925	90	1	cash	2017-03-03 17:3
2	3	44	861	144	1	cash	2017-03-14 4:2
3	4	18	935	156	1	credit_card	2017-03-26 12:4
4	5	18	883	156	1	credit_card	2017-03-01 4:3

```
print(data.isnull().values.any())
data.describe()
```

```
False
```

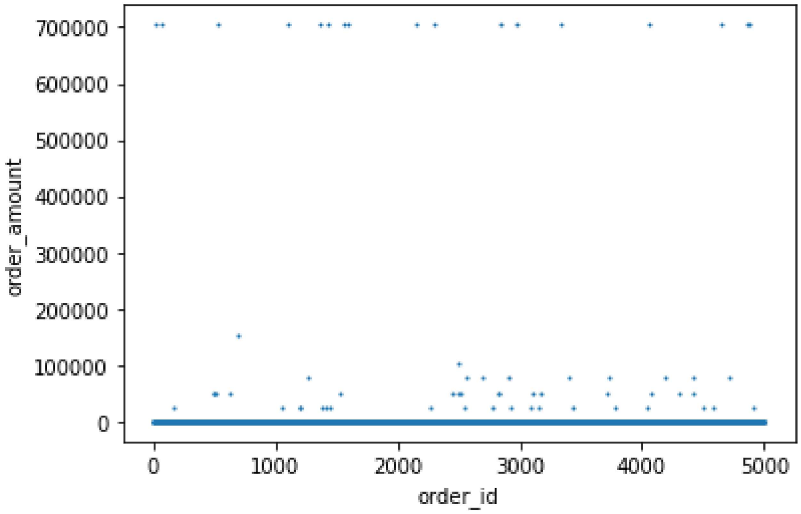
	order_id	shop_id	user_id	order_amount	total_items
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720
std	1443.520003	29.006118	87.798982	41282.539349	116.32032
min	1.000000	1.000000	607.000000	90.000000	1.00000
25%	1250.750000	24.000000	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

- It can be observed right away that the Average Order Value (AOV) mentioned in the questions is simply the mean/average value of all the order amounts.
- Another interesting insight is that the standar deviation for the order amount is extremely high which indicates a wide range of order amount values.
- Similarly, for the total items column, there is a huge variance in the value - (8.79-116.32, 8.79+116.32)
- The range of order amounts is (90,704000) and the range for total items is (1,2000)

Visualize distribution of Order Amounts

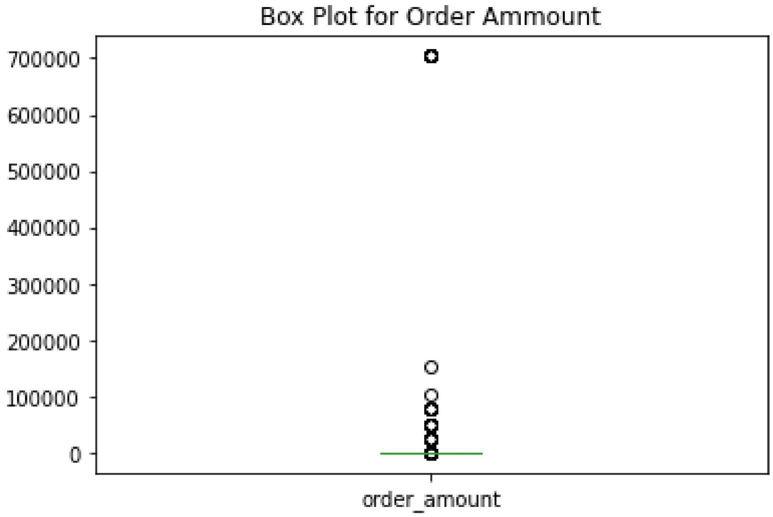
```
data.plot.scatter('order_id','order_amount',s=1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98160a5310>
```



```
# data.boxplot('order_amount')
data['order_amount'].plot.box(title="Box Plot for Order Ammount")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9814165310>
```



Outliers

```
from matplotlib.cbook import boxplot_stats
boxplot_stats(data['order_amount'])
```

```
[{'cihi': 289.04011571494146,
  'cilo': 278.95988428505854,
  'fliers': array([704000, 704000,    780,    765,  25725,    780,    765,    780,
            780,  51450,  51450,  51450, 704000,    830,  51450,    748,
         154350,    772,    804,    815,    885,   1056,    784,  25725,
         704000,    815,    885,  25725,  25725,    935,  77175, 704000,
           1760,   1408,  25725,  25725, 704000,  25725,   1408,    765,
            736,  51450, 704000,    960, 704000,    800,    804,    800,
            865,    745,    830,    880,    920,    765,    774,    790,
            784, 704000,  25725, 704000,    948,    845,    760,    745,
          51450, 102900,    965,  51450,  51450,  25725,    935,  77175,
            780,  77175,    805,  25725,  51450,  51450, 704000,  77175,
          25725,    830, 704000,   1056,    890,    980,  25725,  51450,
            760,  25725,  51450,    748,    786, 704000,  77175,    736,
            805,  25725,   1056,    736,    935,   1086,    736,  51450,
          77175,  25725,    816,    810,    740,  25725, 704000,  51450,
           1064,  77175,    780,  51450,  51450,  77175,    735,  25725,
            760,    880,    780,    748,    748,  25725,    748,    800,
         704000,    780,  77175,    960, 704000,    790, 704000,    760,
          25725,    765,    880,    865,    772]),
  'iqr': 227.0,
  'mean': 3145.128,
  'med': 284.0,
  'q1': 163.0,
  'q3': 390.0,
  'whishi': 730,
  'whislo': 90}]
```

- The interquantile range is 227.0
- q1 = 163.0
- q2 = 390
- whisker high = 730.0

- From the above 2 plots it can be clearly observed that the order amount column has a lot of outliers which is the reason for high standard deviation.
- The issue with naively calculating the average order value is that it does not take into account the number of items sold in each order. Moreover, along with the bulk orders there are orders in the dataset where only one but expensive items is bought which further increases the average value.
- There are 2 ways to deal with this:
  - Firstly, we can calculate the AOV by ignoring the outliers.
  - Second, we can use the median metric to track the AOV.

Mean

```
# mean with all the data points included, original data
data['order_amount'].mean()
```

```
3145.128
```

```
# create modified data frame by excluding the outliers
data2 = data[data['order_amount']<=730.0]
```

```
# mean without outliers
# the whishi value will work as the upper limit to cut off the outliers

print('Number of Outliers =>',len(data[data['order_amount']>730.0]))
print('Mean excluding outliers =>',data2['order_amount'].mean())
```

```
Number of Outliers => 141
Mean excluding outliers => 293.7153735336489
```

Median Value

```
# original median
data['order_amount'].median()
```

```
     284.0
```

```
# median excluding outliers
print("Median excluding outliers =>",data2['order_amount'].median())
```

```
     Median excluding outliers => 280.0
```

- It would be a good idea to use Median as a Metric to track the average order value since it is much more robust that the mean to outliers.