Fall 2022 Data Science Intern Challenge

By: Abhishek Kathuria

Please complete the following questions, and provide your thought process/work. You can attach your work in a text file, link, etc. on the application page. Please ensure answers are easily visible for reviewers!

Question 1: Given some sample data, write a program to answer the following: click here to access the required data set

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.

We know that the Average order value (AOV) calculates the average spent every time a customer places an sneaker order.

Formula:
AOV = sum(order_amount)/ sum(total_items)

Problem: Using Exploratory Data Analysis and Python libraries such as Pandas, Matplotlib, and Seaborn (which is explained in detail below), I identified the which was an outlier problem. Earlier, AOV was being calculated as the mean of the 'order_amount' column. But It's value $3145.12) for a 30 day period was very high for a sneaker store. The explained code shows that an outlier with shop_id = 78 was commiting a fraud by charging $25725 for every pair of sneakers. Hence, the metrics which was selected as mean to calculate AOV was not appropriate as it was being affected by the values of outliers.

Better way to evaluate data: I presented two ways to evaluate the data after handling outlier. The first was either removing outliers and re-calculating the mean of 'order_amount' column as AOV. However it wouldn't be an optimized approach as removing data (even outliers) can prove to be risky.
Another approach which I would personally choose is using the **median** as the AOV or Average Order Value. Since it is the middle value of the sorted values, it is a central value but large and small outliers do not skew the median as much as the mean.

B) What metric would you report for this dataset?

My preferred metric would be to use the Median for the 'order_amount' column as AOV.

C) What is its value?

Approach 1: Removing outliers and re-calculating AOV -> $302.5
Approach 2: Using Median value as AOV -> $284.0 (my preferred approach)

Explanation for Question 1

Import Libraries and Load dataset

In [77]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


df=pd.read_csv("Dataset.csv")
print('Shape of Data ->',df.shape,'\n')
df.head()
```

```
Shape of Data -> (5000, 7)
```

Out[77]:

	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

Exploratory Data Analysis or EDA

In [78]:
```
# Check for Nulls and Duplicates
df.info()
print('\n\n', 'Checking for duplicates')
print(df[df.duplicated()])
```

```
<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


 Checking for duplicates
Empty DataFrame
Columns: [order_id, shop_id, user_id, order_amount, total_items, payment_metho
d, created_at]
Index: []
```

Observation : We can see that there are no null values and duplicates in the data. Hence, our data looks clean!

In [79]:
```
# Looking Closely at the order_amount column
df['order_amount'].describe()
```

Out[79]:
```
count      5000.000000
mean       3145.128000
std       41282.539349
min          90.000000
25%         163.000000
50%         284.000000
75%         390.000000
max      704000.000000
Name: order_amount, dtype: float64
```

Observation:

- We notice that the mean of the 'order_amount' column is $3145.13 (which is the calculated AOV) which is very high for a shoe store. Hence we can conclude that some **outliers** might be affecting it's value.
- Also, Standard Deviation is also very large with it's value of $41282.53. This means that data points are far from mean and therefore for calculating AOV, mean is not a good statistic.

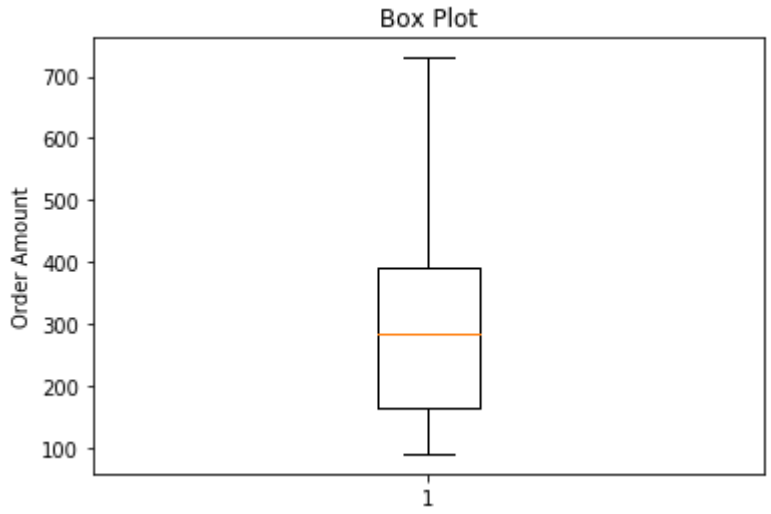
- Given that AOV is $3145.13, we note that 75% of data is below 390 but the max value is $704000.0 which is very large. Hence AOV might be definitely affected by some outliers to give such a large Average Order Value.

Outlier Detection

1. BoxPlot

In [80]:
```
fig1, ax1 = plt.subplots()
ax1.set_title('Box Plot')
ax1.boxplot(df.order_amount,showfliers=False)
ax1.set_ylabel('Order Amount')
```

Out[80]:
```
Text(0, 0.5, 'Order Amount')
```



Observation: From boxplot it is visible that the maximum amount is very large as compared to the amount for the 3rd quartile hence there are visible chances of outliers.

2. Price for each item

We check the price for each item to check if there is some descepancy in the amount charged for a particular shoe.

In [96]:
```
df['price_per_item'] = df['order_amount']/df['total_items']
df
```

Out[96]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	price_pe
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	
...	...	...	...	...	...	...	...	
4995	4996	73	993	330	2	debit	2017-03-30 13:47:17	
4996	4997	48	789	234	2	cash	2017-03-16 20:36:16	
4997	4998	56	867	351	3	cash	2017-03-19 5:42:42	
4998	4999	60	825	354	2	credit_card	2017-03-16 14:51:18	
4999	5000	44	734	288	2	debit	2017-03-18 15:48:18	

5000 rows × 8 columns

In [95]:
```
a= list(df['price_per_item'].unique())
a= sorted(a,reverse=True)
print('Top 3 prices for each individual pair of sneakers')
print('Prices ->', a[:3],'\n')
print('The shop_id of the Outlier (the shop which is charging $25725.0 for a sing
print('Shop Id of Outlier->  ',df.loc[df['price_per_item']==25725.0].shop_id.unic
```

```
Top 3 prices for each individual pair of sneakers
Prices -> [25725.0, 352.0, 201.0]

The shop_id of the Outlier (the shop which is charging $25725.0 for a single pa
ir of sneakers!):
Shop Id of Outlier->   [78]
```

Observation : Yes! We have successfully detected the outlier using Exploratory Data Analysis and Boxplot.
The Shop_id =78 is a fraud as every price_per_item was less than $395 that means every pair of sneakers sold was less than the value of $395 except for one shop which sold at $25,725 which is extraordinarily high!

Outlier Handling

1. Remove outliers and re-calculate AOV again
2. Use Median as a metric

My approach : I think both approaches are correct. Therefore, after removing outliers and re-calculating the mean of 'order_amount' column or AOV, the value comes to be **$302.5.**
On the other hand, If I would use median as the AOV or Average Order Value, the value comes out to be **$284.0.** Since it is the middle value of the sorted values, it is a central value but large and small outliers do not skew the median as much as the mean.

In [98]:
```
# Removing Outliers
new_df= df[df.order_amount<25725]
new_df.describe()
```

Out[98]:

	order_id	shop_id	user_id	order_amount	total_items	price_per_item
count	4937.000000	4937.000000	4937.000000	4937.000000	4937.000000	4937.000000
mean	2499.551347	49.846465	849.752279	302.580514	1.994734	151.788536
std	1444.069407	29.061131	86.840313	160.804912	0.982821	29.034215
min	1.000000	1.000000	700.000000	90.000000	1.000000	90.000000
25%	1248.000000	24.000000	775.000000	163.000000	1.000000	132.000000
50%	2497.000000	50.000000	850.000000	284.000000	2.000000	153.000000
75%	3751.000000	74.000000	925.000000	387.000000	3.000000	166.000000
max	5000.000000	100.000000	999.000000	1760.000000	8.000000	352.000000

In [104]:
```
new_AOV=new_df['order_amount'].mean()

print (" The average order value by removing outliers is: $", new_AOV)
```

```
 The average order value by removing outliers is: $ 302.58051448247926
```

In [85]:
```
# Calculating median of order amount
print('The median is : $', df.order_amount.median())
```

```
The median is : $ 284.0
```

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(*) FROM Orders
JOIN Shippers ON Shippers.ShipperID = Orders.ShipperID
WHERE Shippers.ShipperName = 'Speedy Express';
```

Answer: 54

B) What is the last name of the employee with the most orders?

```
SELECT Employees.LastName, Count() AS num_orders FROM Orders
JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
GROUP BY Employees.LastName ORDER BY num_orders DESC;
```

Answer: Peacock (number of orders are 40)

C) What product was ordered the most by customers in Germany?

```
SELECT Products.ProductName, Sum(OrderDetails.Quantity) as most_ordered FROM Orders
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
WHERE Country = 'Germany'
GROUP BY Products.ProductName ORDER BY most_ordered DESC;
```

Answer: Boston Crab Meat (with 160 orders)

In [ ]: