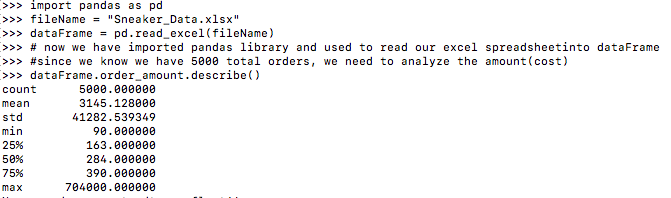
**Name:** Karam Chaban **Date**: 15 – 01 – 2022

**Data Science Intern Challenge Assessment Report**

1. **Sneaker AOV Analysis**

a) Before analyzing the data using python and the panda library, I sorted the order amount descendingly. The top 63 amounts were above 25000 whether it was due to a wholesale order, a heavily overpriced sneaker or a fraudulent error it definitely skewed the numbers. The remaining 4937 orders were all under 1800 so we have outliers that contribute to the AOV being extremely high. Not only that, but the first 17 orders can easily be seen to be the exact same order to the same user from the same shop for a total of 34000 shoes all purchased in the same month. Not only does this heavily skew the number but definitely should be verified for fraudulent error. Now let’s take a better look at the order amount number for a better analysis.

**Figure 1: Description of spreadsheet**

Now we could plot the information for a better visualization but in my professional opinion I think the issue with AOV is clear. Although the mean matches our AOV it is clear we need to analyze this data differently as the AOV does not reflect most stores. 75% of the orders are 390 and under. With 41282.54 as the standard deviation, we can see that there is massive variance, and that this data is unreliable.

b) If we want to select a metric for the AOV for the entire dataset then I believe the median is the best choice. The median is the middle number (50%) which can be seen as 284.

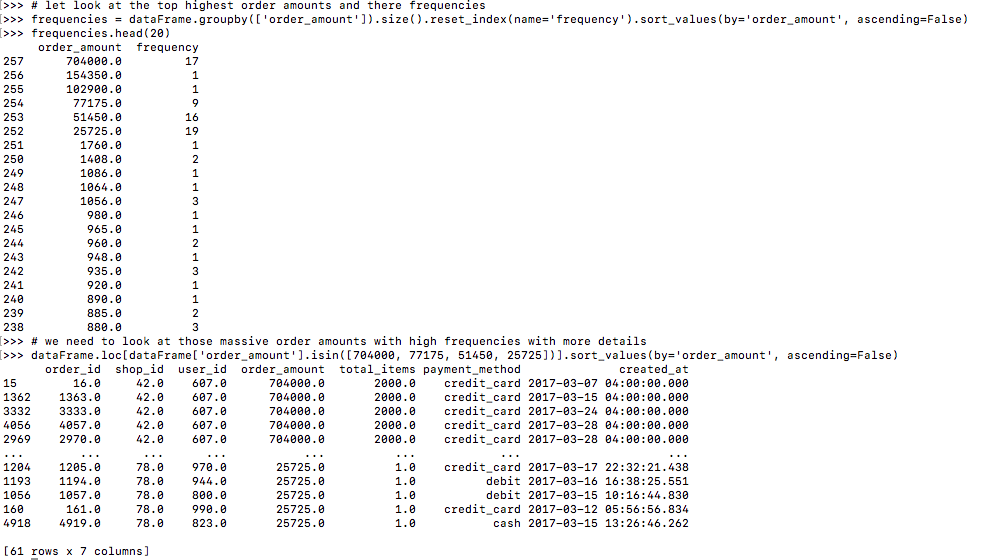
c) 284.

Graphical user interface, text, email

Description automatically generated**Additional Information**: A commonly used method to remove outliers is done by getting the IQR multiplying it by 1.5 and then adding it to q3 for limit above and subtracting it from q1 for the limit below. IQR = 75%- 25% = 227. Without even calculating the lower limit we can see that IQR is greater than Q1(25%) and hence our lower limit is in the negative and that we should include our min which is 90 as part of our AOV. Now for our upper limit, Q3(75%) + 1.5 \* IQR -> 390 + 340.5 =730.5. 1 simple panda command repeated can help us truncate our data.

**Figure2: Values for given percentiles**

This leads me to believe that we should calculate the AOV for the first 97% of our data as they fall between our outlier limits. We now need to look at the top frequencies with highest order amounts.



**Figure 3: Order Amount Frequency Table| Table based on highest frequencies with high total amounts**

Remember what I said at the start. 63 amounts are extreme outliers and were easily shown by sorting the excel sheet. Shop id 42 is responsible for line 257 in our frequencies table while shop 78 is responsible for 256 to 252 as the numbers are clear multiples of 25725. We should calculate the AOV for each of those shops independently.

Calculate the AOV for the order amounts from 97% percentile which is above 712 based on figure 2 until 1768 (at line 251 based on figure 3) as 1. The remaining 97% should have their own unique AOV calculated which leaves with 4 AOVs that would represent their respective stores accurately.

2. **SQL Queries**

a)

SELECT Shippers.ShipperName, COUNT(Orders.ShipperID) AS TotalShips FROM Orders

JOIN Shippers

ON Orders.ShipperID = Shippers.ShipperID

WHERE Shippers.ShipperName = "Speedy Express"

The result is 54. We require data that is dependent on 2 tables that share a key which is ShipperID. We Join both tables based on equating said key. We use the COUNT function to count the frequency of appearances of ShipperID based on the condition where the name is “Speedy Express”. SELECT is then used to display a clear representation of the shipper name and how many orders it shipped.

b)

SELECT Orders.EmployeeID, Employees.LastName, COUNT(Orders.EmployeeID)

AS TotalOrders

FROM Employees

JOIN Orders

ON Employees.EmployeeID = Orders.EmployeeID

GROUP BY 1,2

ORDER BY 3 DESC

The result is Peacock. Here we need to retrieve the employee’s last name with the most orders by joining the employee table and orders table on the shared key, EmployeeID. We can then retrieve the last name based on employee ID and the total number of orders by counting the frequencies of employee id in the Orders table. Group By is used to distinguish the frequency orders from different employees based on their ID and name which is the first and second column respectively, and we order descendingly based on the third column which puts the employee last name with most orders at the top. We could use Limit 1 to just return the top row, but I like to be able to compare my data. The employee ID is not asked for, but I prefer to show it as it is a key and hence ensure the correctness of the data being displayed.

c)

SELECT Products.ProductID, Products.ProductName, SUM( OrderDetails.Quantity) AS TotalOrdered

FROM Customers

JOIN Orders

ON Customers.CustomerID = Orders.CustomerID

JOIN OrderDetails

ON Orders.OrderID = OrderDetails.OrderID

JOIN Products

ON OrderDetails.ProductID = Products.ProductID

WHERE Country = "Germany"

Group By 1,2

Order By 3 DESC

The answer is Boston Crab Meat. Here we need to access data from 4 different tables based on 3 unique keys, a transitive property if you will. Each table joins with the next table based on a different key. This means the tables in between must have at least 2 keys, 1 for the prior table and 1 for the next. We join the tables on the required shared keys. We place a condition on the country being “Germany '', notice we do not need to specify the Customers table as the only other table, Suppliers, that contains a country column is not included in our join. We then retrieve the product’s ID, product’s name and the sum of the quantity which is distinct for each product as we group by the first and second column. We order by the 3rd column descendingly which contains the total ordered for each product and places the highest at the top. Similar comments as above to why we display the key and return the entire table instead of just the first row.