shopify_dsci_2021_fall

January 20, 2022

```
[1]: import pandas as pd
```

```
[2]: df = pd.read_csv("shopify_sneakers.csv")
```

0.1 Getting Familiar with the Data

```
[3]: df.head()
```

```
[3]:    order_id  shop_id  user_id  order_amount  total_items payment_method  \
     0         1       53      746           224            2           cash
     1         2       92      925            90            1           cash
     2         3       44      861           144            1           cash
     3         4       18      935           156            1    credit_card
     4         5       18      883           156            1    credit_card

                 created_at
     0  2017-03-13 12:36:56
     1  2017-03-03 17:38:52
     2   2017-03-14 4:23:56
     3  2017-03-26 12:43:37
     4   2017-03-01 4:35:11
```

```
[4]: df.dtypes
```

```
[4]: order_id           int64
     shop_id            int64
     user_id            int64
     order_amount       int64
     total_items        int64
     payment_method    object
     created_at        object
     dtype: object
```

This is some minor data cleaning and to just confirm that the data set is in a 30-day period

```
[5]: df["created_at"] = pd.to_datetime(df['created_at'])
     dateSorted = df.sort_values(by = ["created_at"])
```

```python
dateSorted.head()
```

[5]:

```
      order_id  shop_id  user_id  order_amount  total_items payment_method  \
1862      1863       39      738           536            4           cash
1741      1742       39      910           268            2           cash
3228      3229       97      912           324            2           cash
1267      1268       80      798           290            2    credit_card
2689      2690       49      799           258            2    credit_card

              created_at
1862 2017-03-01 00:08:09
1741 2017-03-01 00:10:19
3228 2017-03-01 00:14:12
1267 2017-03-01 00:19:31
2689 2017-03-01 00:22:25
```

[6]:

```python
print("First\n", dateSorted.iloc[0])
print("Last\n", dateSorted.iloc[len(df) - 1])
```

```
First
 order_id                         1863
shop_id                            39
user_id                           738
order_amount                      536
total_items                         4
payment_method                   cash
created_at        2017-03-01 00:08:09
Name: 1862, dtype: object
Last
 order_id                         2458
shop_id                            95
user_id                           700
order_amount                      168
total_items                         1
payment_method            credit_card
created_at        2017-03-30 23:55:35
Name: 2457, dtype: object
```

0.2 Question 1a

Assuming the AOV is a simple calculation of: $\frac{total\_amount}{total\_orders}$

Then this would probably just be a mean of the `order_amount` column.

[7]:

```python
df["order_amount"].mean()
```

[7]:

```
3145.128
```

As outlined in the challenge, this is much higher than expected if we consider the fact that we are dealing with sneaker shops. If we take a look at the dataset itself to see what is going on:

```
[8]: df["order_amount"].max()
```

```
[8]: 704000
```

The AOV is so high because there is a purchase order worth $704,000!

```
[9]: df.iloc[df["order_amount"].argmax()]
```

```
[9]: order_id                           16
     shop_id                            42
     user_id                           607
     order_amount                   704000
     total_items                      2000
     payment_method            credit_card
     created_at        2017-03-07 04:00:00
     Name: 15, dtype: object
```

And now we understand why, this looks to be a bulk order of some kind, and assuming there are several of these in the data set, this is why the AOV is skewed higher than expected.

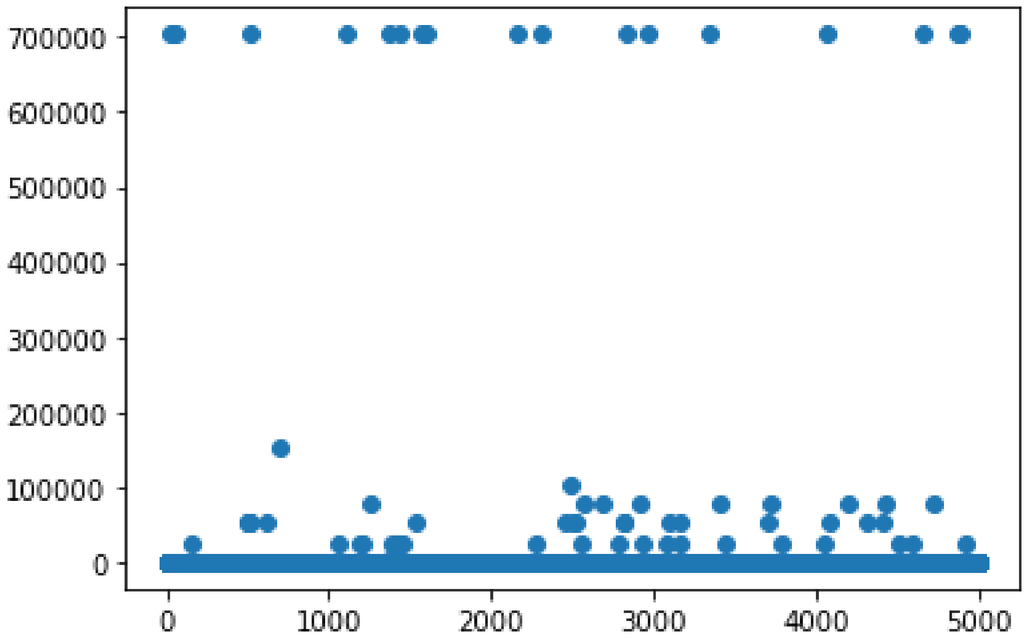
0.3 Questions 1b and 1c

The problem with using an average or a mean calculation in data analysis, is that they are easily infleunced by outliers. In the context of our problem, an outlier could be the example above: a bulk order for 2000 sneakers. Rather than using a mean, I would recommend taking a trimmed mean instead, which can reduce the influence from the extreme outliers we have in our data but still hopefully calculate a reasonable AOV.

The question however is what percentage to trim by.

```
[10]: import matplotlib.pyplot as plt
      plt.scatter(df.order_id, df.order_amount)
```

```
[10]: <matplotlib.collections.PathCollection at 0x7f50d3456d60>
```



I count roughly 30 or so values that seem much too high to be normal from a visual glance which is around 0.6% however just to be a bit more assured, we'll trim by 1% instead.

```
[11]: from scipy import stats

      amounts = df.order_amount
      percentage = 1/100
      stats.trim_mean(amounts, percentage).round(2)
```

```
[11]: 372.16
```

With a slight adjustment to the AOV calculation, we now have a more reasonable value of $372.16

0.3.1 Question 2a

```
[ ]: """
     SELECT COUNT(OrderID) FROM [Orders]
         WHERE ShipperID = (SELECT ShipperID FROM [Shippers]
                              WHERE ShipperName = "Speedy Express")

     ANSWER: 54
     """
```

0.3.2 Question 2b

[ ]:

```
"""
SELECT LastName FROM [Employees]
    WHERE EmployeeID = (SELECT EmployeeID FROM [Orders]
                            GROUP BY EmployeeID
                            ORDER BY COUNT(*) DESC
                            LIMIT 1)

ANSWER: Peacock
"""
```

0.3.3 Question 2c

[ ]:

```
"""
SELECT ProductName FROM [Products]
    WHERE ProductID = (SELECT ProductID FROM [OrderDetails]
                            WHERE OrderID IN (SELECT OrderID FROM [Orders]
                                                  WHERE CustomerID IN (SELECT␣
 ↪CustomerID FROM [Customers]
                                                                                     ␣
 ↪WHERE Country = "Germany"))
                            GROUP BY ProductID
                            ORDER BY SUM(Quantity) DESC
                            LIMIT 1)

ANSWER: Boston Crab Meat
"""
```