Summer 2022 Data Science Challenge

Question 1:

- A) : Upon first glance, an amount of $3145.13 is very unrealistic for the expected value of an average Sneaker order, let's investigate:

```
#Necessary Imports
import numpy as py
import pandas as pd
import matplotlib as pl

data=pd.read_csv("C:\\Users\\Yassi\\OneDrive\\Bureau\\shopifyFiles\\shopifySpreadsheet.csv")
print(data.head()) #printing sample to ensure proper setup
pd.options.display.max_columns=len(data.columns)
pd.options.display.width=None
skewedData=data[data['total_items']==2000] #sample set data for clear outliers
print(skewedData)
totalIncomingFunds , sampleSize = data.order_amount.sum(),data.order_amount.count()
print("Data Set Mean/AOV: ", totalIncomingFunds/sampleSize)
```

After making the necessary imports (had issues on my interpreter with regards to plotting data, will not be included in report) and reading the csv info from our data set, we run a few tests to check on possible issues, and come to the following conclusions:

- When filtering by maximum order amount for each total item count occurrence, we notice two red flags:
    - Many of the results are multiples of 25725, which may indicate an exploit or a bug. Not only that, but we come to realize that these orders are all from Shop 78.
    - Whilst the total item counts are generally reasonable, our data set reflects the presence of one or more orders with 2000 items, which is a cause for concern as these may be orders placed by scripts in order to clear out stock, or a potential huge bug. As well, these orders are all made by the same user (607), at the same shop (42), at the same time (4:00 A.M).

```
typicalSample=data[data['total_items']<=10]
potentialError=typicalSample[typicalSample['order_amount']>1000]
print(potentialError)
```

160	161	78	990	25725	1	credit_card	2017-03-12 5:56:57
490	491	78	936	51450	2	debit	2017-03-26 17:08:19
493	494	78	983	51450	2	cash	2017-03-16 21:39:35
511	512	78	967	51450	2	cash	2017-03-09 7:23:14
617	618	78	760	51450	2	cash	2017-03-18 11:18:42
691	692	78	878	154350	6	debit	2017-03-27 22:51:43
938	939	42	808	1056	3	credit_card	2017-03-13 23:43:45
1056	1057	78	800	25725	1	debit	2017-03-15 10:16:45
1193	1194	78	944	25725	1	debit	2017-03-16 16:38:26
1204	1205	78	970	25725	1	credit_card	2017-03-17 22:32:21
1259	1260	78	775	77175	3	credit_card	2017-03-27 9:27:20
1364	1365	42	797	1760	5	cash	2017-03-10 6:28:21
1367	1368	42	926	1408	4	cash	2017-03-13 2:38:34
1384	1385	78	867	25725	1	cash	2017-03-17 16:38:06
1419	1420	78	912	25725	1	cash	2017-03-30 12:23:43
1452	1453	78	812	25725	1	credit_card	2017-03-17 18:09:54
1471	1472	42	907	1408	4	debit	2017-03-12 23:00:22
1529	1530	78	810	51450	2	cash	2017-03-29 7:12:01
2270	2271	78	855	25725	1	credit_card	2017-03-14 23:58:22
2452	2453	78	709	51450	2	cash	2017-03-27 11:04:04
2492	2493	78	834	102900	4	debit	2017-03-04 4:37:34
2495	2496	78	707	51450	2	cash	2017-03-26 4:38:52
2512	2513	78	935	51450	2	debit	2017-03-18 18:57:13
2548	2549	78	861	25725	1	cash	2017-03-17 19:36:00
2564	2565	78	915	77175	3	debit	2017-03-25 1:19:35
2690	2691	78	962	77175	3	debit	2017-03-22 7:33:25
2773	2774	78	890	25725	1	cash	2017-03-26 10:36:43
2818	2819	78	869	51450	2	debit	2017-03-17 6:25:51
2821	2822	78	814	51450	2	cash	2017-03-02 17:13:25
2906	2907	78	817	77175	3	debit	2017-03-16 3:45:46
2922	2923	78	740	25725	1	debit	2017-03-12 20:10:58
2987	2988	42	819	1056	3	cash	2017-03-03 9:09:25
3085	3086	78	910	25725	1	cash	2017-03-26 1:59:27
3101	3102	78	855	51450	2	credit_card	2017-03-21 5:10:34
3151	3152	78	745	25725	1	credit_card	2017-03-18 13:13:07
3167	3168	78	927	51450	2	cash	2017-03-12 12:23:08
3403	3404	78	928	77175	3	debit	2017-03-16 9:45:05
3440	3441	78	982	25725	1	debit	2017-03-19 19:02:54
3513	3514	42	726	1056	3	debit	2017-03-24 17:51:05
3538	3539	43	830	1086	6	debit	2017-03-17 19:56:29
3705	3706	78	828	51450	2	credit_card	2017-03-14 20:43:15
3724	3725	78	766	77175	3	credit_card	2017-03-16 14:13:26
3780	3781	78	889	25725	1	cash	2017-03-11 21:14:50
4040	4041	78	852	25725	1	cash	2017-03-02 14:31:12
4079	4080	78	946	51450	2	cash	2017-03-20 21:14:00
4141	4142	54	733	1064	8	debit	2017-03-07 17:05:18
4192	4193	78	787	77175	3	credit_card	2017-03-18 9:25:32
4311	4312	78	960	51450	2	debit	2017-03-01 3:02:10
4412	4413	78	756	51450	2	debit	2017-03-02 4:13:39
4420	4421	78	969	77175	3	debit	2017-03-09 15:21:35
4505	4506	78	866	25725	1	debit	2017-03-22 22:06:01
4584	4585	78	997	25725	1	cash	2017-03-25 21:48:44
4715	4716	78	818	77175	3	debit	2017-03-05 5:10:44
4918	4919	78	823	25725	1	cash	2017-03-15 13:26:46

```
Shop 78 average price per item:  25725.0
```

We firstly conclude that shop 78 constitutes an outlier, whether voluntarily or not, due to this deviation.

Next:

```
      order_id  shop_id  user_id  order_amount  total_items payment_method          created_at
15          16       42      607        704000         2000    credit_card  2017-03-07 4:00:00
60          61       42      607        704000         2000    credit_card  2017-03-04 4:00:00
520        521       42      607        704000         2000    credit_card  2017-03-02 4:00:00
1104      1105       42      607        704000         2000    credit_card  2017-03-24 4:00:00
1362      1363       42      607        704000         2000    credit_card  2017-03-15 4:00:00
1436      1437       42      607        704000         2000    credit_card  2017-03-11 4:00:00
1562      1563       42      607        704000         2000    credit_card  2017-03-19 4:00:00
1602      1603       42      607        704000         2000    credit_card  2017-03-17 4:00:00
2153      2154       42      607        704000         2000    credit_card  2017-03-12 4:00:00
2297      2298       42      607        704000         2000    credit_card  2017-03-07 4:00:00
2835      2836       42      607        704000         2000    credit_card  2017-03-28 4:00:00
2969      2970       42      607        704000         2000    credit_card  2017-03-28 4:00:00
3332      3333       42      607        704000         2000    credit_card  2017-03-24 4:00:00
4056      4057       42      607        704000         2000    credit_card  2017-03-28 4:00:00
4646      4647       42      607        704000         2000    credit_card  2017-03-02 4:00:00
4868      4869       42      607        704000         2000    credit_card  2017-03-22 4:00:00
4882      4883       42      607        704000         2000    credit_card  2017-03-25 4:00:00
Data Set Mean/AOV:  3145.128
```

We may also interpret these results as potential fraud or bugs as stated previously.

In sum, these two outliers, from a potential few others, skew the mean value of our data set far too much for us to consider the AOV a viable metric to report.

B) We could compensate for these errors by calculating a more precise average (sum the order amounts, divide by the sum of total items in Data Set). However, I propose that the median would be a more accurate metric to use in this case. If we are to interpret the order amount column as a sorted array, the value sitting at the middle point would constitute a much better metric for us to evaluate the data set from, as it would minimize the impact outliers like those we saw have on our analysis. Indeed, our new result when calculating the median is much closer to the expectations one would have around a Sneaker shop's typical transaction amount:

```
Data Set Mean/AOV:  3145.128
Median value in Data Set:  284.0
```

C) As calculated previously, the code block:

```
15    print("Median value in Data Set: ", data.order_amount.median())
```

Will output our Median value, which is $284.

Question 2:

A) Query:

```
1   //Query 1
2   SELECT COUNT(*)
3   FROM Orders AS item,  Shippers AS shipment
4   WHERE ShipperName="Speedy Express" AND item.ShipperId=shipment.ShipperId
5   //
6   Answer: 54
```

Result: 54.

B) Query:

```
 8   //Query 2
 9   SELECT emp.LastName, COUNT(*) AS Count
10   FROM Orders AS item, Employees AS emp
11   WHERE item.EmployeeID = emp.EmployeeID
12   GROUP BY item.EmployeeID
13   ORDER BY Count DESC
14   LIMIT 1;
15   //
16   Answer: Peacock, 40 Orders
```

Answer: Peacock.

C) Query:

```
//Query 3
SELECT prod.ProductID,prod.ProductName
FROM Orders AS item, OrderDetails AS det, Customers AS cust, Products AS prod
WHERE det.OrderID=item.OrderID AND cust.Country="Germany" AND det.ProductID=prod.ProductID AND cust.CustomerID=item.CustomerID
GROUP BY prod.ProductID
ORDER BY SUM(Quantity) DESC
LIMIT 1;
//
```

Answer: Boston Crab Meat, ID 40.