Shopify DS Application Q1ABC

January 11, 2022

0.1 Question 1.A

[16]:
```
import pandas as pd
import numpy as np
import math
from statistics import mean

sneaker = pd.read_csv('2019 Winter Data Science Intern Challenge Data Set -␣
 ↪Sheet1.csv')
sneaker
```

[16]:
```
      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
…          …       …       …          …           …            …
4995      4996       73      993           330            2          debit
4996      4997       48      789           234            2           cash
4997      4998       56      867           351            3           cash
4998      4999       60      825           354            2    credit_card
4999      5000       44      734           288            2          debit

               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
…                    …
4995  2017-03-30 13:47:17
4996  2017-03-16 20:36:16
4997   2017-03-19 5:42:42
4998  2017-03-16 14:51:18
4999  2017-03-18 15:48:18

[5000 rows x 7 columns]
```

[26]:
```
summary = sneaker.describe()
summary
```

[26]:

	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

[93]:
```
sneaker['order_amount'].median()
```

[93]: 284.0

Checking for any missing values

[22]:
```
sneaker.isnull().sum()
```

[22]:
```
order_id          0
shop_id           0
user_id           0
order_amount      0
total_items       0
payment_method    0
created_at        0
dtype: int64
```

We can conclude that there are no missing(null) values

To figure out where the value $3145.13, came from (as given in the question), I assumed that it was just taken from the mean of the order amounts. I came up with this assumption from observing that mean of order_amount in the summary chart above, matched the value given. Since the values match exactly, I can safely assume that this was how the value of $3145.13 was actually calculated. To be sure:

[19]:
```
pred_aov = statistics.mean(sneaker.order_amount)
pred_aov
```

[19]: 3145.128

We know that the average order value that was calculated is $3145.13, which doesn't make sense as sneakers (on average) are not nearly as expensive. However, we can divide the total revenue by the number of orders for each shop (by grouping by shop_id), and then obtain the average for all of these values.

[46]:
```
statistics.mean(sneaker.groupby('shop_id').apply(lambda x:
    (sum(x['order_amount']))/(sum(x['total_items']))))
```

[46]: 407.99

This is still a high average for affordable sneakers. This leads me to believe that there are some outliers which have a very high price, that is driving the mean to be higher than expected. Let's see what the boxplot looks like:

[47]:
```
sneaker.boxplot(column = "order_amount")
```

[47]: <AxesSubplot:>

There are a lot of outliers! Also referring back to the summary chart, I can see that the maximum value for the order_amount is $704000, this is INSANELY high! There is also a standard deviation of about 41282, which means there is a lot of variation within the data points in the order amount column. When thinking about the types of buyers that could exist, there are buyers that are just buying for themselves, or there are store owners/suppliers buying a bulk amount of shoes, which explains the high values order amounts. Let's dig into this issue:

[54]:
```
outlier = sneaker.groupby(['order_amount']).size().reset_index(name =
    'count_of_order_amount').sort_values(by='order_amount', ascending = False)
outlier.head(15)
```

[54]:

	order_amount	count_of_order_amount
257	704000	17

```
256         154350                      1
255         102900                      1
254          77175                      9
253          51450                     16
252          25725                     19
251           1760                      1
250           1408                      2
249           1086                      1
248           1064                      1
247           1056                      3
246            980                      1
245            965                      1
244            960                      2
243            948                      1
```

This chart gives us a visual of how there are several orders of $704000, in fact 17 orders, we can also see that the same pattern follows for order amounts of $51450, $25725 with many orders with the exact same amounts.

[55]:
```python
big_order_amounts = sneaker.sort_values(by= 'order_amount', ascending = False)
big_order_amounts.head(15)
```

[55]:
```
      order_id  shop_id  user_id  order_amount  total_items payment_method  \
2153      2154       42      607        704000         2000    credit_card
3332      3333       42      607        704000         2000    credit_card
520        521       42      607        704000         2000    credit_card
1602      1603       42      607        704000         2000    credit_card
60          61       42      607        704000         2000    credit_card
2835      2836       42      607        704000         2000    credit_card
4646      4647       42      607        704000         2000    credit_card
2297      2298       42      607        704000         2000    credit_card
1436      1437       42      607        704000         2000    credit_card
4882      4883       42      607        704000         2000    credit_card
4056      4057       42      607        704000         2000    credit_card
15          16       42      607        704000         2000    credit_card
1104      1105       42      607        704000         2000    credit_card
1562      1563       42      607        704000         2000    credit_card
2969      2970       42      607        704000         2000    credit_card

             created_at  new_aov
2153  2017-03-12 4:00:00    352.0
3332  2017-03-24 4:00:00    352.0
520   2017-03-02 4:00:00    352.0
1602  2017-03-17 4:00:00    352.0
60    2017-03-04 4:00:00    352.0
2835  2017-03-28 4:00:00    352.0
4646  2017-03-02 4:00:00    352.0
```

2297	2017-03-07 4:00:00	352.0
1436	2017-03-11 4:00:00	352.0
4882	2017-03-25 4:00:00	352.0
4056	2017-03-28 4:00:00	352.0
15	2017-03-07 4:00:00	352.0
1104	2017-03-24 4:00:00	352.0
1562	2017-03-19 4:00:00	352.0
2969	2017-03-28 4:00:00	352.0

From the table above, it looks like 1 shop (shop_id is 42) has all the big orders of $704000. Let us check if this is the case for $51450 and $25725

[59]:
```
sneaker.loc[sneaker['order_amount'].isin([704000, 51450, 25725])].
↪sort_values(by='order_amount', ascending=False)
```

[59]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	\
15	16	42	607	704000	2000	credit_card	
1362	1363	42	607	704000	2000	credit_card	
2969	2970	42	607	704000	2000	credit_card	
2835	2836	42	607	704000	2000	credit_card	
4056	4057	42	607	704000	2000	credit_card	
60	61	42	607	704000	2000	credit_card	
2297	2298	42	607	704000	2000	credit_card	
2153	2154	42	607	704000	2000	credit_card	
1562	1563	42	607	704000	2000	credit_card	
1436	1437	42	607	704000	2000	credit_card	
1602	1603	42	607	704000	2000	credit_card	
3332	3333	42	607	704000	2000	credit_card	
1104	1105	42	607	704000	2000	credit_card	
4882	4883	42	607	704000	2000	credit_card	
4868	4869	42	607	704000	2000	credit_card	
520	521	42	607	704000	2000	credit_card	
4646	4647	42	607	704000	2000	credit_card	
511	512	78	967	51450	2	cash	
3167	3168	78	927	51450	2	cash	
3705	3706	78	828	51450	2	credit_card	
3101	3102	78	855	51450	2	credit_card	
490	491	78	936	51450	2	debit	
2821	2822	78	814	51450	2	cash	
2818	2819	78	869	51450	2	debit	
493	494	78	983	51450	2	cash	
2495	2496	78	707	51450	2	cash	
2512	2513	78	935	51450	2	debit	
2452	2453	78	709	51450	2	cash	
4079	4080	78	946	51450	2	cash	
617	618	78	760	51450	2	cash	
1529	1530	78	810	51450	2	cash	

4311	4312	78	960	51450	2	debit
4412	4413	78	756	51450	2	debit
3440	3441	78	982	25725	1	debit
4040	4041	78	852	25725	1	cash
3780	3781	78	889	25725	1	cash
4505	4506	78	866	25725	1	debit
4584	4585	78	997	25725	1	cash
2548	2549	78	861	25725	1	cash
3151	3152	78	745	25725	1	credit_card
3085	3086	78	910	25725	1	cash
2922	2923	78	740	25725	1	debit
2773	2774	78	890	25725	1	cash
2270	2271	78	855	25725	1	credit_card
1452	1453	78	812	25725	1	credit_card
1419	1420	78	912	25725	1	cash
1384	1385	78	867	25725	1	cash
1204	1205	78	970	25725	1	credit_card
1193	1194	78	944	25725	1	debit
1056	1057	78	800	25725	1	debit
160	161	78	990	25725	1	credit_card
4918	4919	78	823	25725	1	cash

	created_at	new_aov
15	2017-03-07 4:00:00	352.0
1362	2017-03-15 4:00:00	352.0
2969	2017-03-28 4:00:00	352.0
2835	2017-03-28 4:00:00	352.0
4056	2017-03-28 4:00:00	352.0
60	2017-03-04 4:00:00	352.0
2297	2017-03-07 4:00:00	352.0
2153	2017-03-12 4:00:00	352.0
1562	2017-03-19 4:00:00	352.0
1436	2017-03-11 4:00:00	352.0
1602	2017-03-17 4:00:00	352.0
3332	2017-03-24 4:00:00	352.0
1104	2017-03-24 4:00:00	352.0
4882	2017-03-25 4:00:00	352.0
4868	2017-03-22 4:00:00	352.0
520	2017-03-02 4:00:00	352.0
4646	2017-03-02 4:00:00	352.0
511	2017-03-09 7:23:14	25725.0
3167	2017-03-12 12:23:08	25725.0
3705	2017-03-14 20:43:15	25725.0
3101	2017-03-21 5:10:34	25725.0
490	2017-03-26 17:08:19	25725.0
2821	2017-03-02 17:13:25	25725.0
2818	2017-03-17 6:25:51	25725.0

```
493   2017-03-16 21:39:35  25725.0
2495   2017-03-26 4:38:52  25725.0
2512  2017-03-18 18:57:13  25725.0
2452  2017-03-27 11:04:04  25725.0
4079  2017-03-20 21:14:00  25725.0
617   2017-03-18 11:18:42  25725.0
1529   2017-03-29 7:12:01  25725.0
4311   2017-03-01 3:02:10  25725.0
4412   2017-03-02 4:13:39  25725.0
3440  2017-03-19 19:02:54  25725.0
4040  2017-03-02 14:31:12  25725.0
3780  2017-03-11 21:14:50  25725.0
4505  2017-03-22 22:06:01  25725.0
4584  2017-03-25 21:48:44  25725.0
2548  2017-03-17 19:36:00  25725.0
3151  2017-03-18 13:13:07  25725.0
3085   2017-03-26 1:59:27  25725.0
2922  2017-03-12 20:10:58  25725.0
2773  2017-03-26 10:36:43  25725.0
2270  2017-03-14 23:58:22  25725.0
1452  2017-03-17 18:09:54  25725.0
1419  2017-03-30 12:23:43  25725.0
1384  2017-03-17 16:38:06  25725.0
1204  2017-03-17 22:32:21  25725.0
1193  2017-03-16 16:38:26  25725.0
1056  2017-03-15 10:16:45  25725.0
160    2017-03-12 5:56:57  25725.0
4918  2017-03-15 13:26:46  25725.0
```

We can observe from the table above that the same pattern occurs, however for order amounts of \$704000, there are 2000 total items, but for order amounts of \$51450, there are 2 total items, and for order amounts of \$25725, there is only 1 total item. Since double \$25725 is exactly \$51450, this leads me to believe that the orders with \$51450, are just 2 orders of \$25725. I can also conclude that the orders with order amounts of \$704000 could not be justified as a regular buyer (customers only buying a reasonable amount of pairs of shoes at a reasonable price), as it probably is a supplier buying mass amounts of shoes (hence, the 2000 orders).

However the orders of \$25725 seem to be very high if it is just one pair of shoe, I will assume that there was a data entry error for this, and assume it was inputted as cents instead of dollars. I will change both values of \$25725 and \$51450 as dollar amounts, since both order amounts relate to one another.

[87]:
```
sneaker.loc[sneaker['order_amount'] == 51450, 'order_amount'] = 514.50
sneaker.loc[sneaker['order_amount'] == 25725, 'order_amount'] = 257.25
```

[89]:
```
sneaker.loc[sneaker['order_amount'].isin([704000, 514.50, 257.25])].
↪sort_values(by='order_amount', ascending=False)
```

[89]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method \
15	16	42	607	704000.00	2000	credit_card
1362	1363	42	607	704000.00	2000	credit_card
2969	2970	42	607	704000.00	2000	credit_card
2835	2836	42	607	704000.00	2000	credit_card
4056	4057	42	607	704000.00	2000	credit_card
60	61	42	607	704000.00	2000	credit_card
2297	2298	42	607	704000.00	2000	credit_card
2153	2154	42	607	704000.00	2000	credit_card
1562	1563	42	607	704000.00	2000	credit_card
1436	1437	42	607	704000.00	2000	credit_card
1602	1603	42	607	704000.00	2000	credit_card
3332	3333	42	607	704000.00	2000	credit_card
1104	1105	42	607	704000.00	2000	credit_card
4882	4883	42	607	704000.00	2000	credit_card
4868	4869	42	607	704000.00	2000	credit_card
520	521	42	607	704000.00	2000	credit_card
4646	4647	42	607	704000.00	2000	credit_card
511	512	78	967	514.50	2	cash
3167	3168	78	927	514.50	2	cash
3705	3706	78	828	514.50	2	credit_card
3101	3102	78	855	514.50	2	credit_card
490	491	78	936	514.50	2	debit
2821	2822	78	814	514.50	2	cash
2818	2819	78	869	514.50	2	debit
493	494	78	983	514.50	2	cash
2495	2496	78	707	514.50	2	cash
2512	2513	78	935	514.50	2	debit
2452	2453	78	709	514.50	2	cash
4079	4080	78	946	514.50	2	cash
617	618	78	760	514.50	2	cash
1529	1530	78	810	514.50	2	cash
4311	4312	78	960	514.50	2	debit
4412	4413	78	756	514.50	2	debit
3440	3441	78	982	257.25	1	debit
4040	4041	78	852	257.25	1	cash
3780	3781	78	889	257.25	1	cash
4505	4506	78	866	257.25	1	debit
4584	4585	78	997	257.25	1	cash
2548	2549	78	861	257.25	1	cash
3151	3152	78	745	257.25	1	credit_card
3085	3086	78	910	257.25	1	cash
2922	2923	78	740	257.25	1	debit
2773	2774	78	890	257.25	1	cash
2270	2271	78	855	257.25	1	credit_card
1452	1453	78	812	257.25	1	credit_card
1419	1420	78	912	257.25	1	cash

1384	1385	78	867	257.25	1	cash
1204	1205	78	970	257.25	1	credit_card
1193	1194	78	944	257.25	1	debit
1056	1057	78	800	257.25	1	debit
160	161	78	990	257.25	1	credit_card
4918	4919	78	823	257.25	1	cash

	created_at	new_aov
15	2017-03-07 4:00:00	352.0
1362	2017-03-15 4:00:00	352.0
2969	2017-03-28 4:00:00	352.0
2835	2017-03-28 4:00:00	352.0
4056	2017-03-28 4:00:00	352.0
60	2017-03-04 4:00:00	352.0
2297	2017-03-07 4:00:00	352.0
2153	2017-03-12 4:00:00	352.0
1562	2017-03-19 4:00:00	352.0
1436	2017-03-11 4:00:00	352.0
1602	2017-03-17 4:00:00	352.0
3332	2017-03-24 4:00:00	352.0
1104	2017-03-24 4:00:00	352.0
4882	2017-03-25 4:00:00	352.0
4868	2017-03-22 4:00:00	352.0
520	2017-03-02 4:00:00	352.0
4646	2017-03-02 4:00:00	352.0
511	2017-03-09 7:23:14	25725.0
3167	2017-03-12 12:23:08	25725.0
3705	2017-03-14 20:43:15	25725.0
3101	2017-03-21 5:10:34	25725.0
490	2017-03-26 17:08:19	25725.0
2821	2017-03-02 17:13:25	25725.0
2818	2017-03-17 6:25:51	25725.0
493	2017-03-16 21:39:35	25725.0
2495	2017-03-26 4:38:52	25725.0
2512	2017-03-18 18:57:13	25725.0
2452	2017-03-27 11:04:04	25725.0
4079	2017-03-20 21:14:00	25725.0
617	2017-03-18 11:18:42	25725.0
1529	2017-03-29 7:12:01	25725.0
4311	2017-03-01 3:02:10	25725.0
4412	2017-03-02 4:13:39	25725.0
3440	2017-03-19 19:02:54	25725.0
4040	2017-03-02 14:31:12	25725.0
3780	2017-03-11 21:14:50	25725.0
4505	2017-03-22 22:06:01	25725.0
4584	2017-03-25 21:48:44	25725.0
2548	2017-03-17 19:36:00	25725.0

```
3151  2017-03-18 13:13:07  25725.0
3085   2017-03-26 1:59:27  25725.0
2922  2017-03-12 20:10:58  25725.0
2773  2017-03-26 10:36:43  25725.0
2270  2017-03-14 23:58:22  25725.0
1452  2017-03-17 18:09:54  25725.0
1419  2017-03-30 12:23:43  25725.0
1384  2017-03-17 16:38:06  25725.0
1204  2017-03-17 22:32:21  25725.0
1193  2017-03-16 16:38:26  25725.0
1056  2017-03-15 10:16:45  25725.0
160    2017-03-12 5:56:57  25725.0
4918  2017-03-15 13:26:46  25725.0
```

[90]:
```
statistics.mean(sneaker.groupby('shop_id').apply(lambda x:␣
↪(sum(x['order_amount']))/(sum(x['total_items']))))
```

[90]:
```
260.3928125
```

An AOV of $260.39 seems a lot better!

0.2 Question 1. B

I would report the median for this dataset, since we saw that one shop (shop_id 42) has 17 transactions of 2000 units with the value of \$704000 from seller 42 (seller_id), which skews the data very heavily. Median values are not as heavily influenced by outliers as opposed to the mean.

0.3 Question 1. C

It's value is $284.0 as analyzed above.