Fall 2021 DS Intern Challenge

May 7, 2021

1 Question 1

Given some sample data, write a program to answer the following: click here to access the required data set (https://docs.google.com/spreadsheets/d/16i38oonuX1y1g7C_UAmiK9GkY7cS-64DfiDMNiR41LM/edit#gid=0)

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.

Think about what could be going wrong with our calculation. Think about a better way to evaluate this data. What metric would you report for this dataset? What is its value?

[119]:
```python
import pandas as pd
import numpy as np
```

[120]:
```python
sales_df = pd.read_csv("sales_data.csv")
```

[121]:
```python
sales_df.info()
```

```
<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
```

First, I will take a look at the dataset with some samples to get a rough idea of the structure of the dataset.

[122]:
```
#visualize the data
sales_df.sample(20)
```

[122]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	\
1940	1941	46	916	166	1	debit	
2379	2380	24	980	280	2	debit	
2584	2585	89	700	392	2	cash	
2300	2301	82	764	531	3	credit_card	
3176	3177	96	856	459	3	debit	
1544	1545	34	794	122	1	credit_card	
3249	3250	81	941	354	2	debit	
1315	1316	86	763	260	2	cash	
1312	1313	82	927	531	3	credit_card	
2484	2485	8	996	528	4	debit	
79	80	20	838	254	2	credit_card	
1686	1687	74	901	459	3	credit_card	
3627	3628	16	860	312	2	debit	
4565	4566	40	782	161	1	cash	
942	943	93	915	456	4	cash	
4608	4609	82	743	354	2	cash	
1085	1086	7	970	224	2	debit	
1613	1614	18	792	156	1	credit_card	
3813	3814	46	813	498	3	cash	
3883	3884	60	957	354	2	debit	

	created_at
1940	2017-03-26 19:16:48
2379	2017-03-08 18:46:33
2584	2017-03-21 12:18:52
2300	2017-03-27 23:29:38
3176	2017-03-01 7:47:32
1544	2017-03-24 11:41:02
3249	2017-03-11 14:09:22
1315	2017-03-06 1:32:05
1312	2017-03-20 8:32:00
2484	2017-03-09 17:26:20
79	2017-03-03 14:00:25
1686	2017-03-01 0:43:39
3627	2017-03-22 20:07:42
4565	2017-03-21 3:39:41
942	2017-03-12 7:00:39
4608	2017-03-13 10:30:55
1085	2017-03-08 16:54:11
1613	2017-03-10 7:37:09
3813	2017-03-27 3:31:57
3883	2017-03-29 4:19:59

1.1 Naive AOV

Given the data are already in a 30 day window (March), the naive way to calculate AOV is to divide the revenue by the number of orders.

[123]:
```
Naive_AOV_30 = sales_df.order_amount.sum() / sales_df.order_id.count()
```

[124]:
```
Naive_AOV_30
```

[124]:
```
3145.128
```

However, if we look at the 20 samples from above, none of the order_amount value are close to \$3145.13. To further investigate, we shall look at the highest order_amount in the data, and see if there is skewness.

[75]:
```
sales_df.skew(axis=0)
```

[75]:
```
order_id          0.000000
shop_id           0.013830
user_id          -0.034052
order_amount     16.675033
total_items      17.065556
dtype: float64
```

Both the order_amount and total_items are pretty positively skewed, we shall take a look the top order_amount and total_items.

[125]:
```
sales_df.sort_values(['order_amount','total_items'], ascending=False).head(50)
```

[125]:
```
      order_id  shop_id  user_id  order_amount  total_items payment_method  \
15          16       42      607        704000         2000    credit_card
60          61       42      607        704000         2000    credit_card
520        521       42      607        704000         2000    credit_card
1104      1105       42      607        704000         2000    credit_card
1362      1363       42      607        704000         2000    credit_card
1436      1437       42      607        704000         2000    credit_card
1562      1563       42      607        704000         2000    credit_card
1602      1603       42      607        704000         2000    credit_card
2153      2154       42      607        704000         2000    credit_card
2297      2298       42      607        704000         2000    credit_card
2835      2836       42      607        704000         2000    credit_card
2969      2970       42      607        704000         2000    credit_card
3332      3333       42      607        704000         2000    credit_card
4056      4057       42      607        704000         2000    credit_card
4646      4647       42      607        704000         2000    credit_card
4868      4869       42      607        704000         2000    credit_card
4882      4883       42      607        704000         2000    credit_card
691        692       78      878        154350            6          debit
2492      2493       78      834        102900            4          debit
```

1259	1260	78	775	77175	3	credit_card
2564	2565	78	915	77175	3	debit
2690	2691	78	962	77175	3	debit
2906	2907	78	817	77175	3	debit
3403	3404	78	928	77175	3	debit
3724	3725	78	766	77175	3	credit_card
4192	4193	78	787	77175	3	credit_card
4420	4421	78	969	77175	3	debit
4715	4716	78	818	77175	3	debit
490	491	78	936	51450	2	debit
493	494	78	983	51450	2	cash
511	512	78	967	51450	2	cash
617	618	78	760	51450	2	cash
1529	1530	78	810	51450	2	cash
2452	2453	78	709	51450	2	cash
2495	2496	78	707	51450	2	cash
2512	2513	78	935	51450	2	debit
2818	2819	78	869	51450	2	debit
2821	2822	78	814	51450	2	cash
3101	3102	78	855	51450	2	credit_card
3167	3168	78	927	51450	2	cash
3705	3706	78	828	51450	2	credit_card
4079	4080	78	946	51450	2	cash
4311	4312	78	960	51450	2	debit
4412	4413	78	756	51450	2	debit
160	161	78	990	25725	1	credit_card
1056	1057	78	800	25725	1	debit
1193	1194	78	944	25725	1	debit
1204	1205	78	970	25725	1	credit_card
1384	1385	78	867	25725	1	cash
1419	1420	78	912	25725	1	cash

	created_at
15	2017-03-07 4:00:00
60	2017-03-04 4:00:00
520	2017-03-02 4:00:00
1104	2017-03-24 4:00:00
1362	2017-03-15 4:00:00
1436	2017-03-11 4:00:00
1562	2017-03-19 4:00:00
1602	2017-03-17 4:00:00
2153	2017-03-12 4:00:00
2297	2017-03-07 4:00:00
2835	2017-03-28 4:00:00
2969	2017-03-28 4:00:00
3332	2017-03-24 4:00:00
4056	2017-03-28 4:00:00

```
4646    2017-03-02 4:00:00
4868    2017-03-22 4:00:00
4882    2017-03-25 4:00:00
691    2017-03-27 22:51:43
2492    2017-03-04 4:37:34
1259    2017-03-27 9:27:20
2564    2017-03-25 1:19:35
2690    2017-03-22 7:33:25
2906    2017-03-16 3:45:46
3403    2017-03-16 9:45:05
3724   2017-03-16 14:13:26
4192    2017-03-18 9:25:32
4420   2017-03-09 15:21:35
4715    2017-03-05 5:10:44
490    2017-03-26 17:08:19
493    2017-03-16 21:39:35
511     2017-03-09 7:23:14
617    2017-03-18 11:18:42
1529    2017-03-29 7:12:01
2452   2017-03-27 11:04:04
2495    2017-03-26 4:38:52
2512   2017-03-18 18:57:13
2818    2017-03-17 6:25:51
2821   2017-03-02 17:13:25
3101    2017-03-21 5:10:34
3167   2017-03-12 12:23:08
3705   2017-03-14 20:43:15
4079   2017-03-20 21:14:00
4311    2017-03-01 3:02:10
4412    2017-03-02 4:13:39
160     2017-03-12 5:56:57
1056   2017-03-15 10:16:45
1193   2017-03-16 16:38:26
1204   2017-03-17 22:32:21
1384   2017-03-17 16:38:06
1419   2017-03-30 12:23:43
```

Next, just by looking at the data returned, there is one customer user_id 607 which bought 2000 items from the same shop in mutiple orders, while most orders are under 6 items. Also, the price for one sneaker in shop_id 78 is \$25725, which does not make any sense given that a sneaker is a relatively afforable item. This is the reason why our AOV is so high. To obtain a more reasonable AOV, we shall drop the all the outlier, tuples with total_items = 2000, and the tuples where 1 sneaker is not priced reasonable, ie. shop_id 78 with the sneaker priced at \$25725.

[97]:
```
index = sales_df[(sales_df['order_amount'] >= 25725)].index
sales_df.drop(index, inplace = True)
```

[100]:
```
AOV_30 = sales_df.order_amount.sum() / sales_df.order_id.count()
print(AOV_30)
```

```
302.58051448247926
```

After some data cleaning, the new AOV is \$302.58.

1.2 Using median aggregator instead of average

Apart from cleaning the data by dropping some outliers, we can also apply median which return the central stendency for skewed distributions.

[108]:
```
sales_df = pd.read_csv("sales_data.csv")
```

[113]:
```
sales_df.order_amount.median()
```

[113]:
```
284.0
```

With median, we get \$284 even with the skewed data remain in the dataset.

2 Question 2

For this question you'll need to use SQL. Follow this link (https://www.w3schools.com/SQL/TRYSQL.ASP?FILENAME=TRYSQL_SELECT_ALL) 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?

SQL Query: SELECT count(*) FROM Orders LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID WHERE ShipperName='Speedy Express';

54 orders were shipped by Speedy Express.

b. What is the last name of the employee with the most orders?

SQL Query: SELECT TOP 1 count(Employees.EmployeeID) as number_of_orders, LastName FROM Orders LEFT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID GROUP BY Employees.EmployeeID, LastName ORDER BY count(Employees.EmployeeID) DESC;

Peacock is the last name of the employee with the most orders.

c. What product was ordered the most by customers in Germany?

SQL Query: SELECT TOP 1 P.ProductID, ProductName, SUM(Quantity) as Quantity FROM OrderDetails OD, Orders O, Products P WHERE OD.OrderID = O.OrderID and P.ProductID = OD.ProductID and O.CustomerID IN (SELECT CustomerID FROM Customers C WHERE C.Country = 'Germany') GROUP BY P.ProductID, ProductName ORDER BY SUM(Quantity) DESC;

Boston Crab Meat was being ordered the most and have a total sale of 160 units by customers in Germany.

```
[ ]:
```