Shopify Fall 2022 Data Science Intern Challenge

Question 1:
Preview the dataset:

Load the dataset into Jupiter notebook.

```
#load data
url = 'https://docs.google.com/spreadsheets/d/16i38oonuX1y1g7C_UAmiK9GkY7cS-64DfiDMNiR41LM/export?format=csv&gid=0'
data = pd.read_csv(url)
```

Preview the data:

```
data.head()
```

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
0	1	53	746	224	2	cash	2017-03-13 12:36:56
1	2	92	925	90	1	cash	2017-03-03 17:38:52
2	3	44	861	144	1	cash	2017-03-14 4:23:56
3	4	18	935	156	1	credit_card	2017-03-26 12:43:37
4	5	18	883	156	1	credit_card	2017-03-01 4:35:11

Check whether there is any missing value in the dataset.

```
#Check missing value
data.isna().sum()
```

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

Show some basic statistical information about the dataset.

```
#Basic Statistical information
data.describe()
```

	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

Calculate the average order value (AOV)

```
#Calculate AOV
aov = np.mean(data.order_amount)
print('AOV: ', aov)
```

```
AOV:  3145.128
```

a. Think about what could be going wrong with our calculation. Think about a better way to evaluate this data.

From the previous basic statistical information, we can observe that there are outliers in the "order_amount" and "total_item" columns.

Take a look at those order amounts > AOV:

```
#Take a look about those order amount > AOV
data[data.order_amount > aov]
```

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	average
15	16	42	607	704000	2000	credit_card	2017-03-07 4:00:00	352.0
60	61	42	607	704000	2000	credit_card	2017-03-04 4:00:00	352.0
160	161	78	990	25725	1	credit_card	2017-03-12 5:56:57	25725.0
490	491	78	936	51450	2	debit	2017-03-26 17:08:19	25725.0
493	494	78	983	51450	2	cash	2017-03-16 21:39:35	25725.0
...	...	...	...	...	...	...	...	...
4646	4647	42	607	704000	2000	credit_card	2017-03-02 4:00:00	352.0
4715	4716	78	818	77175	3	debit	2017-03-05 5:10:44	25725.0
4868	4869	42	607	704000	2000	credit_card	2017-03-22 4:00:00	352.0
4882	4883	42	607	704000	2000	credit_card	2017-03-25 4:00:00	352.0
4918	4919	78	823	25725	1	cash	2017-03-15 13:26:46	25725.0

63 rows × 8 columns

It shows that most outliers are from shop 42 and shop 78.

```
#Check the outlier shop_id
data[data.order_amount > aov].shop_id.value_counts()
```

```
78    46
42    17
Name: shop_id, dtype: int64
```

As a result, it seems better to split the whole dataset into three groups, shop 42, shop 78, and other shops. Also adding a “average price per pair” column may get a better sight when analyzing.

```
#Curious about average price per pair
#Create a column "average" calculating the average price per pair
data["average"] = data["order_amount"]/data["total_items"]
```

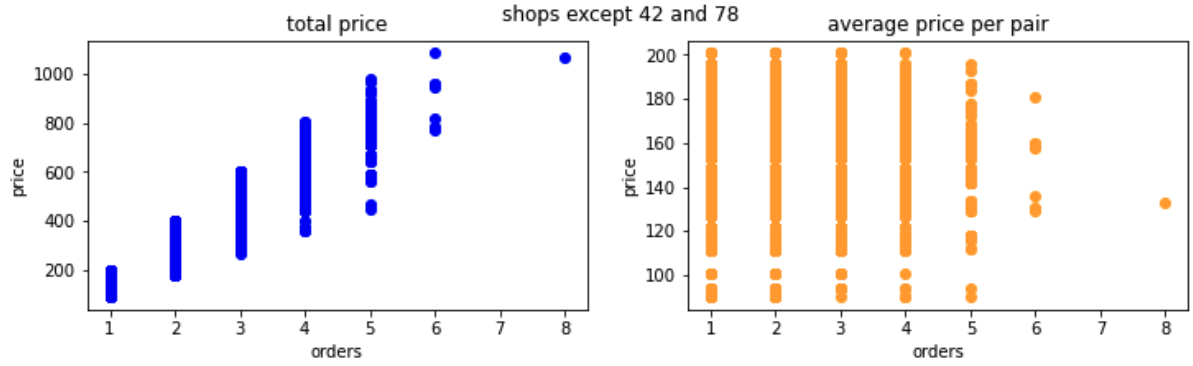
Other Shops:

```
#Other shops basic statistical information
other_data = data[(data.shop_id != 78) & (data.shop_id != 42)]
other_data.describe()
```

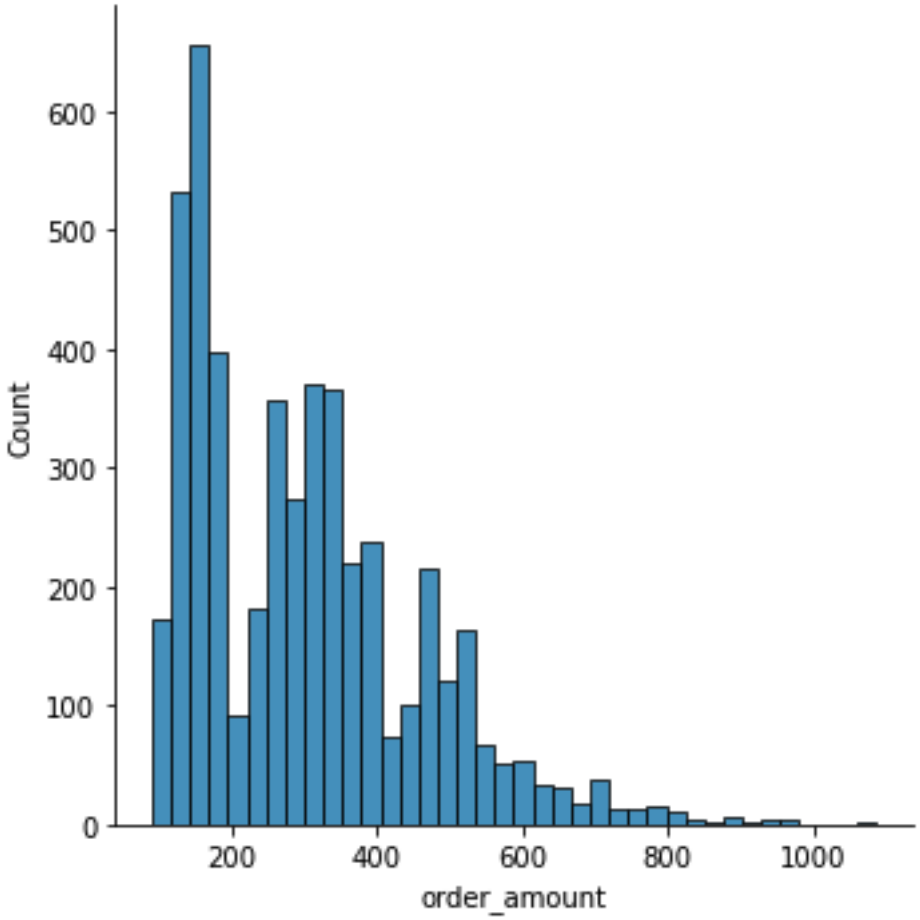
	order_id	shop_id	user_id	order_amount	total_items	average
count	4903.000000	4903.000000	4903.000000	4903.000000	4903.000000	4903.000000
mean	2499.584540	49.900877	849.858862	300.155823	1.995717	150.400163
std	1444.221163	29.154367	86.887947	155.941112	0.982602	23.851202
min	1.000000	1.000000	700.000000	90.000000	1.000000	90.000000
25%	1246.500000	24.000000	776.000000	163.000000	1.000000	132.000000
50%	2499.000000	50.000000	850.000000	284.000000	2.000000	153.000000
75%	3750.500000	74.000000	925.000000	386.500000	3.000000	166.000000
max	5000.000000	100.000000	999.000000	1086.000000	8.000000	201.000000

The basic statistical information looks much more normal than pervious.

Visualize the data:



From these two scatter charts, we can find that there is no big difference in the other shops group. The price seems similar in different shops.

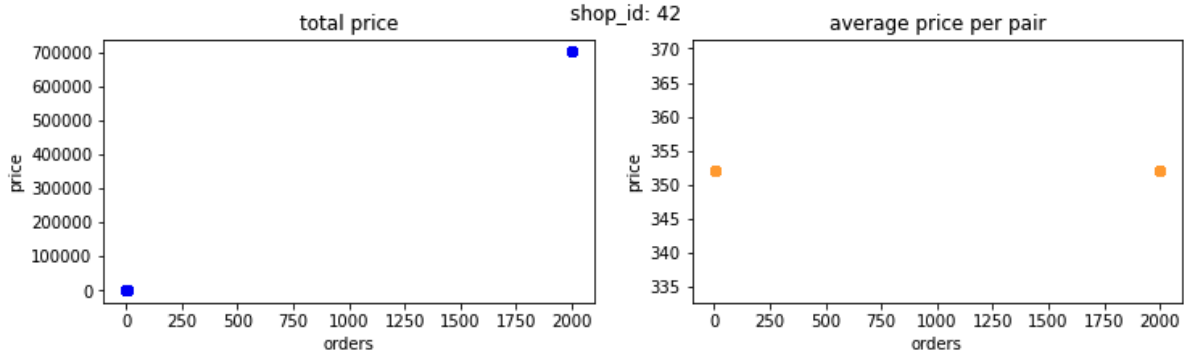


From this histogram, it shows that the order_amount is skewed.

Calculate the AOV, median order value, and skew.

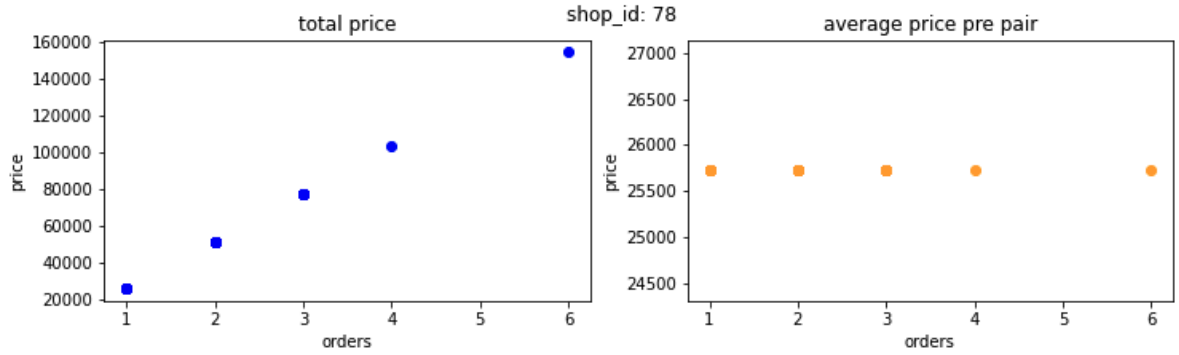
```
Other shops AOV:  300.1558229655313
Other shops median order value:  284.0
Other shops skew:  1.003106067784531
```

Shop 42:



From the scatter charts, they show that the average price per pair is the same no matter how many items the customer buys. Compared with other shops, there are customers who buy extremely large amounts of sneakers at shop 42. What's more, the average price per pair is slightly higher than other shops'.

Shop 78:



From the scatter charts, they show that the difference between shop 78 and other shops is the average price per pair. Shop 78 sells much more expensive sneakers than other shops.

b. What metric would you report for this dataset?

Since we split the dataset into three groups, as a result, we can apply different metrics according to different shops. AOV is a good metric, but not good at the skew dataset. Median order amount and standard deviation are also two great tools to support when evaluating a dataset.

c. What is its value?

shop_type	AOV	Median Order Amount	Standard Deviation	Price Per Pair
42	235101.490196	704	334860.641587	352.000000
78	49213.043478	51450	26472.227449	25725.000000
other	300.155823	284	155.941112	150.400163

Question 2:
a.

```
SELECT count(*) AS Count_Orders FROM Orders
JOIN Shippers USING(ShipperID)
WHERE ShipperName = "Speedy Express";
```

Count_Orders
54

There are 54 orders shipped by Speedy Express in total.

b.

```
With lastnamecount AS(
  SELECT LastName, count(*) AS ln_count FROM orders
  JOIN Employees USING(EmployeeID)
  GROUP BY EmployeeID)

SELECT LastName, Max(ln_count) FROM lastnamecount
```

Number of Records: 1

LastName	Max(ln_count)
Peacock	40

Peacock has the most orders.

c.

```
WITH productcount AS(
  SELECT ProductName, count(*) as p_count FROM Orders
  JOIN Customers USING(CustomerID)
  JOIN OrderDetails USING(OrderID)
  JOIN Products USING(ProductID)
  WHERE Country = "Germany"
  GROUP BY ProductName)

SELECT ProductName, Max(p_count) FROM productcount;
```

Number of Records: 1

ProductName	Max(p_count)
Gorgonzola Telino	5

Most customers in Germany ordered Gorgonzola Telino.