Shopify Summer 2022 Data Science Intern Challenge Q1

Applicant: Ester Tsai

Phone: 925-918-5938

Email: tsaiester@gmail.com

Project Portfolio: https://ester-tsai.github.io/

In [1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Question 1:

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.

Load dataset

In [2]:
```python
data = pd.read_csv("2019 Winter Data Science Intern Challenge Data Set.csv")
data
```

Out[2]:

	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
...	...	...	...	...	...	...	...
4995	4996	73	993	330	2	debit	2017-03-30 13:47:17
4996	4997	48	789	234	2	cash	2017-03-16 20:36:16
4997	4998	56	867	351	3	cash	2017-03-19 5:42:42
4998	4999	60	825	354	2	credit_card	2017-03-16 14:51:18
4999	5000	44	734	288	2	debit	2017-03-18 15:48:18

5000 rows × 7 columns

Check for null values

In [3]:
```python
dataset_contains_null = data.isnull().values.any()
does_or_does_not = "does" if dataset_contains_null else "does not"
reas_or_not_reas = "not reasonable" if dataset_contains_null else "reasonable"
print(f"The dataset {does_or_does_not} contain any null value, so it is {reas_or_not_reas} to proceed o
nward.")
```

```
The dataset does not contain any null value, so it is reasonable to proceed onward.
```

Calculate original AOV

In [4]:
```python
average_order_value = data.get('order_amount').mean()
print(f"Without considering the total_items in each order, \
it appears that the average order value (AOV) is ${round(average_order_value, 2)}")
```

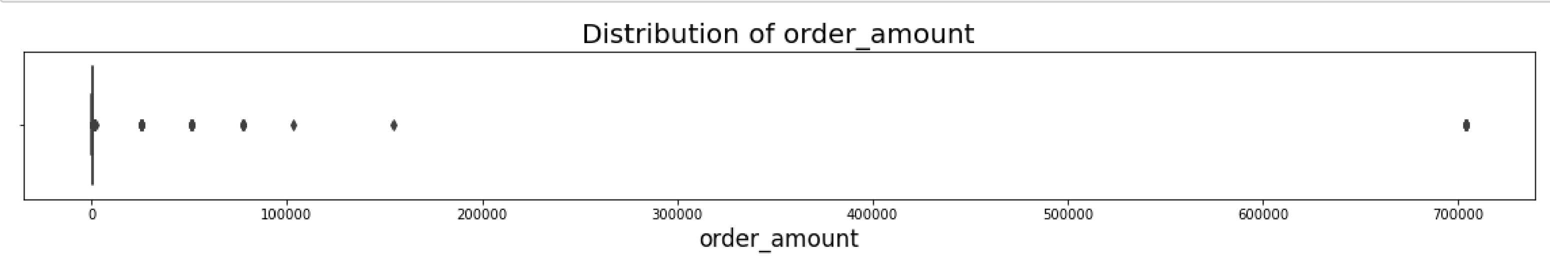
```
Without considering the total_items in each order, it appears that the average order value (AOV) is
$3145.13
```

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

Descriptive data analysis

In [5]:
```python
plt.rcParams["figure.figsize"] = (20,2)
boxplot = sns.boxplot(x=data['order_amount'])
boxplot.axes.set_title("Distribution of order_amount", fontsize=20)
boxplot.set_xlabel("order_amount", fontsize=17)
plt.show()

quantitative_data = data[['order_amount', 'total_items']]
quantitative_data.describe()
```



Out[5]:

	order_amount	total_items
count	5000.000000	5000.00000
mean	3145.128000	8.78720
std	41282.539349	116.32032
min	90.000000	1.00000
25%	163.000000	1.00000
50%	284.000000	2.00000
75%	390.000000	3.00000
max	704000.000000	2000.00000

A boxplot and summary of the dataset reveals that there are several **extreme outliers** with abnormally high *total_items* and *order_amount*.

Identitfy the abnormality

In [6]:
```python
data.sort_values(by='order_amount', ascending=False).head(20)
```

Out[6]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
2153	2154	42	607	704000	2000	credit_card	2017-03-12 4:00:00
3332	3333	42	607	704000	2000	credit_card	2017-03-24 4:00:00
520	521	42	607	704000	2000	credit_card	2017-03-02 4:00:00
1602	1603	42	607	704000	2000	credit_card	2017-03-17 4:00:00
60	61	42	607	704000	2000	credit_card	2017-03-04 4:00:00
2835	2836	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
2297	2298	42	607	704000	2000	credit_card	2017-03-07 4:00:00
1436	1437	42	607	704000	2000	credit_card	2017-03-11 4:00:00
4882	4883	42	607	704000	2000	credit_card	2017-03-25 4:00:00
4056	4057	42	607	704000	2000	credit_card	2017-03-28 4:00:00
15	16	42	607	704000	2000	credit_card	2017-03-07 4:00:00
1104	1105	42	607	704000	2000	credit_card	2017-03-24 4:00:00
1562	1563	42	607	704000	2000	credit_card	2017-03-19 4:00:00
2969	2970	42	607	704000	2000	credit_card	2017-03-28 4:00:00
4868	4869	42	607	704000	2000	credit_card	2017-03-22 4:00:00
1362	1363	42	607	704000	2000	credit_card	2017-03-15 4:00:00
691	692	78	878	154350	6	debit	2017-03-27 22:51:43
2492	2493	78	834	102900	4	debit	2017-03-04 4:37:34
3724	3725	78	766	77175	3	credit_card	2017-03-16 14:13:26

A closer look at the sorted dataframe shows that there is **a significant number of unexpectedly huge orders coming from the same user and same store**. It means that our original AOV was heavily pulled toward the higher end because of these extreme outliers. To reduce the effect these extreme outliars have on our metric, it is crucial to take total_items into consideration.

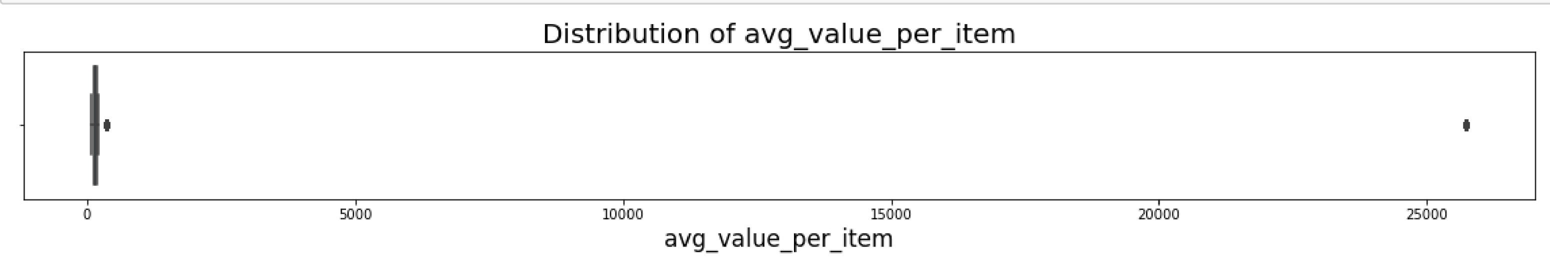
The AOV is so high largely because of orders with more than one total_items. It might help to divide order_amount by total_items for each order, so we can have a better idea how much each item costed on average.

Calculate average value per item

In [7]:
```python
data_with_avg_value_per_item = data.copy(deep=True)
data_with_avg_value_per_item['avg_value_per_item'] = data.get('order_amount') / data.get('total_items')
```

In [8]:
```python
plt.rcParams["figure.figsize"] = (20,2)
boxplot = sns.boxplot(x=data_with_avg_value_per_item['avg_value_per_item'])
boxplot.axes.set_title("Distribution of avg_value_per_item", fontsize=20)
boxplot.set_xlabel("avg_value_per_item", fontsize=17)
plt.show()

data_with_avg_value_per_item[['avg_value_per_item']].describe()
```



Out[8]:

	avg_value_per_item
count	5000.000000
mean	387.742800
std	2441.963725
min	90.000000
25%	133.000000
50%	153.000000
75%	169.000000
max	25725.000000

The bonxplot and table above show that **there is still something peculiar with the average value per item**. The max of $25725 seems abnormally large for a pair of sneakers. So I will take a look at whether these abnormal orders show any patterns.

In [9]:
```python
data_with_avg_value_per_item.sort_values(by='avg_value_per_item', ascending=False).head(50)
```

Out[9]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	avg_value_per_item
4420	4421	78	969	77175	3	debit	2017-03-09 15:21:35	25725.0
2906	2907	78	817	77175	3	debit	2017-03-16 3:45:46	25725.0
4505	4506	78	866	25725	1	debit	2017-03-22 22:06:01	25725.0
493	494	78	983	51450	2	cash	2017-03-16 21:39:35	25725.0
2452	2453	78	709	51450	2	cash	2017-03-27 11:04:04	25725.0
1452	1453	78	812	25725	1	credit_card	2017-03-17 18:09:54	25725.0
3167	3168	78	927	51450	2	cash	2017-03-12 12:23:08	25725.0
3403	3404	78	928	77175	3	debit	2017-03-16 9:45:05	25725.0
4918	4919	78	823	25725	1	cash	2017-03-15 13:26:46	25725.0
2773	2774	78	890	25725	1	cash	2017-03-26 10:36:43	25725.0
2492	2493	78	834	102900	4	debit	2017-03-04 4:37:34	25725.0
2495	2496	78	707	51450	2	cash	2017-03-26 4:38:52	25725.0
1384	1385	78	867	25725	1	cash	2017-03-17 16:38:06	25725.0
4079	4080	78	946	51450	2	cash	2017-03-20 21:14:00	25725.0
3705	3706	78	828	51450	2	credit_card	2017-03-14 20:43:15	25725.0
1259	1260	78	775	77175	3	credit_card	2017-03-27 9:27:20	25725.0
1193	1194	78	944	25725	1	debit	2017-03-16 16:38:26	25725.0
3151	3152	78	745	25725	1	credit_card	2017-03-18 13:13:07	25725.0
2564	2565	78	915	77175	3	debit	2017-03-25 1:19:35	25725.0
1529	1530	78	810	51450	2	cash	2017-03-29 7:12:01	25725.0
3440	3441	78	982	25725	1	debit	2017-03-19 19:02:54	25725.0
4412	4413	78	756	51450	2	debit	2017-03-02 4:13:39	25725.0
2548	2549	78	861	25725	1	cash	2017-03-17 19:36:00	25725.0
2821	2822	78	814	51450	2	cash	2017-03-02 17:13:25	25725.0
4192	4193	78	787	77175	3	credit_card	2017-03-18 9:25:32	25725.0
2512	2513	78	935	51450	2	debit	2017-03-18 18:57:13	25725.0
511	512	78	967	51450	2	cash	2017-03-09 7:23:14	25725.0
2818	2819	78	869	51450	2	debit	2017-03-17 6:25:51	25725.0
490	491	78	936	51450	2	debit	2017-03-26 17:08:19	25725.0
3724	3725	78	766	77175	3	credit_card	2017-03-16 14:13:26	25725.0
1204	1205	78	970	25725	1	credit_card	2017-03-17 22:32:21	25725.0
1419	1420	78	912	25725	1	cash	2017-03-30 12:23:43	25725.0
4311	4312	78	960	51450	2	debit	2017-03-01 3:02:10	25725.0
2690	2691	78	962	77175	3	debit	2017-03-22 7:33:25	25725.0
617	618	78	760	51450	2	cash	2017-03-18 11:18:42	25725.0
3101	3102	78	855	51450	2	credit_card	2017-03-21 5:10:34	25725.0
1056	1057	78	800	25725	1	debit	2017-03-15 10:16:45	25725.0
3780	3781	78	889	25725	1	cash	2017-03-11 21:14:50	25725.0
4584	4585	78	997	25725	1	cash	2017-03-25 21:48:44	25725.0
160	161	78	990	25725	1	credit_card	2017-03-12 5:56:57	25725.0
2922	2923	78	740	25725	1	debit	2017-03-12 20:10:58	25725.0
2270	2271	78	855	25725	1	credit_card	2017-03-14 23:58:22	25725.0
4715	4716	78	818	77175	3	debit	2017-03-05 5:10:44	25725.0
691	692	78	878	154350	6	debit	2017-03-27 22:51:43	25725.0
4040	4041	78	852	25725	1	cash	2017-03-02 14:31:12	25725.0
3085	3086	78	910	25725	1	cash	2017-03-26 1:59:27	25725.0
3697	3698	42	839	352	1	debit	2017-03-12 2:45:09	352.0
1929	1930	42	770	352	1	credit_card	2017-03-17 8:11:13	352.0
1911	1912	42	739	704	2	cash	2017-03-07 5:42:52	352.0
308	309	42	770	352	1	credit_card	2017-03-11 18:14:39	352.0

It appears that all of these abnormally large avg_value_per_item come from the same shop with a shop_id of 78, although with different users.

Part B: What metric would you report for this dataset?

Because of the extreme outliers I identified above, I can consider using **median** as the metric instead of **mean**, so the value I find would be robust to outliers.

It may be helpful to find the median for both AOV and avg_value_per_item, since AOV varies greatly based on total_items.

Part C: What is its value?

In [10]:
```python
median_order_amount = data_with_avg_value_per_item.get('order_amount').median()
print(f"the median order_amount is: ${median_order_amount}")
```

```
the median order_amount is: $284.0
```

In [11]:
```python
median_avg_value_per_item = data_with_avg_value_per_item.get('avg_value_per_item').median()
print(f"the median avg_value_per_item is: ${median_avg_value_per_item}")
```

```
the median avg_value_per_item is: $153.0
```

Q1 Conclusion

The dataset appeared to have an abnormally high AOV of $3145.13 because:

- We did not take into account the effect of total_items on order_amount.
- There were nearly 50 orders with abnormally high average value per item.

Solution:

- Use median instead of mean
- Consider both AOV and avg_value_per_item

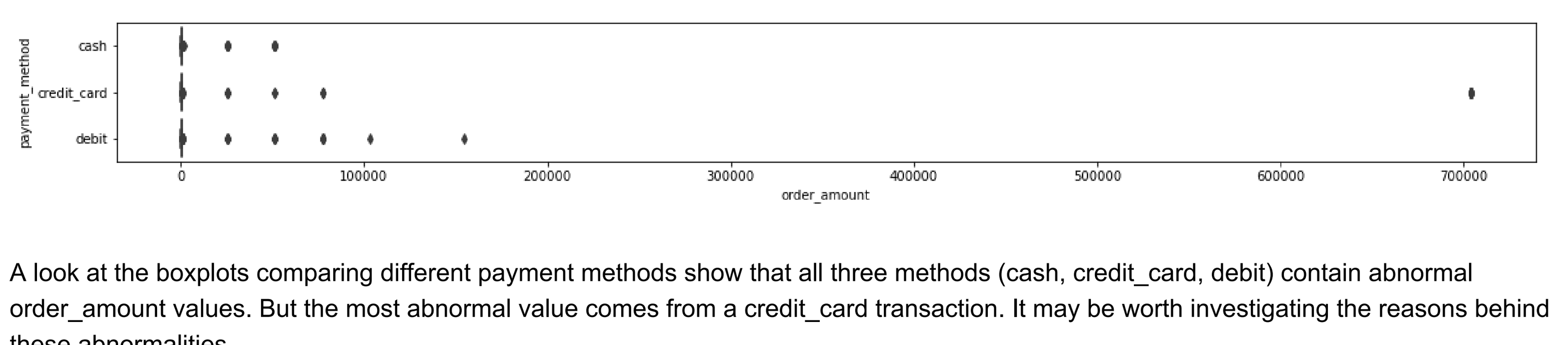
Additional Note (for fun!)

- Please look into shop 78 for potential transaction fraud. All 46 orders at shop 78 have an average value per item of $25725, which is abnormally expensive for a pair of sneakers.

Final Note

In [12]:
```python
sns.boxplot(x=data['order_amount'], y=data['payment_method'])
```

Out[12]:
```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x25a49015ac0&gt;
```



A look at the boxplots comparing different payment methods show that all three methods (cash, credit_card, debit) contain abnormal order_amount values. But the most abnormal value comes from a credit_card transaction. It may be worth investigating the reasons behind these abnormalities.

The end. Thank you for reading my response.