In [1]:
```
#Summer 2022 Data Science Intern Challenge
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
data_frame = pd.read_csv("2019 Winter Data Science Intern Challenge Data Set.csv")
```

In [4]:
```
# a. Think about what could be going wrong with our calculation. Think about a better way to evaluate this data.
```

In [5]:
```
#  HYPOTHESIS
"""
    I think that the AOV = 3145.128000 is wrong because of the following reasons
    1. Outliers might be skewing data
        We can check this by looking at the standard deviation, box plot, comparing max and min order_amount
    2. Bulk Transactions might be skewing data
        a) we need to check the frequency of total_items
        b) We need to check where order_amount is exceptionally high
"""
```

Out[5]:
```
'\n    I think that the AOV = 3145.128000 is wrong because of the following reasons\n    1. Outliers might be skewing data\n        We can check this by looking at the standard deviation, box plot, comparing max and min order_amount\n    2. Bulk Transactions might be skewing data \n        a) we need to check the frequency of total_items\n        b) We need to check where order_amount is exceptionally high\n'
```

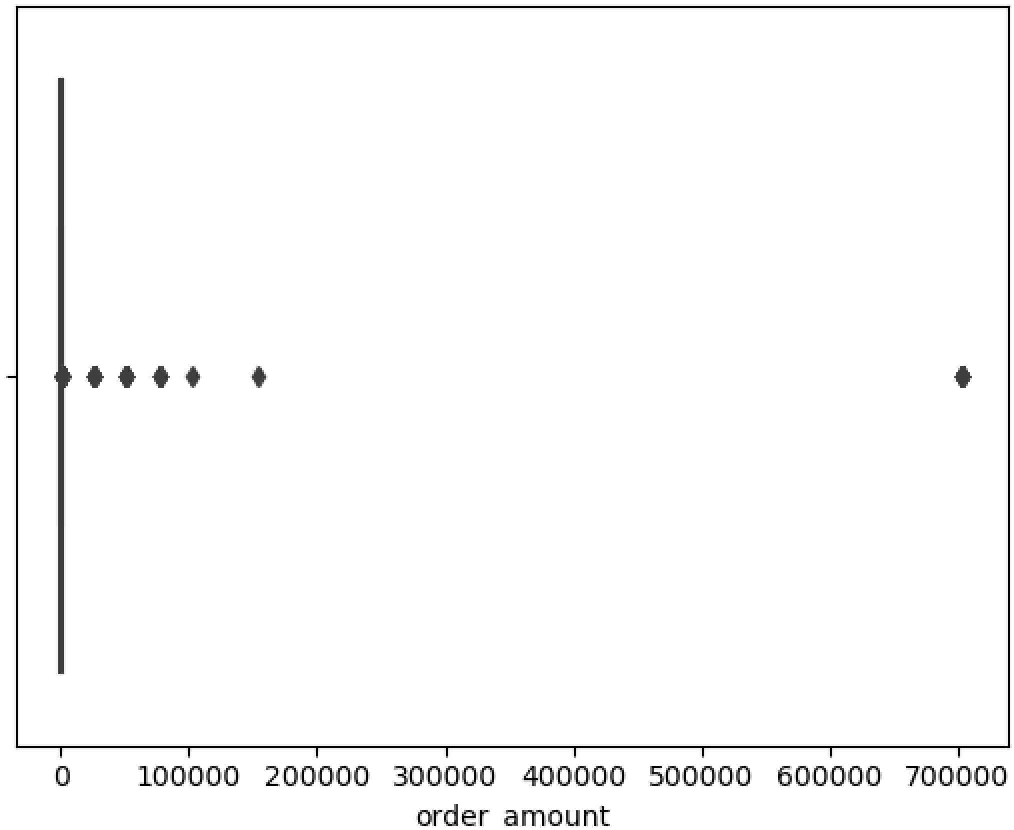
In [6]:
```
print (data_frame.order_amount.describe())
```

```
count      5000.000000
mean       3145.128000
std       41282.539349
min          90.000000
25%         163.000000
50%         284.000000
75%         390.000000
max      704000.000000
Name: order_amount, dtype: float64
```

In [7]:
```
# We can confirm that the wrong calculation for AOV is in fact the mean of order_amount.
# The above data supports hypothesis 1. as the standard deviation is very high and the difference
# between the max and min is also a lot.
```

In [8]:
```
#We can confirm the same with a box plot
```

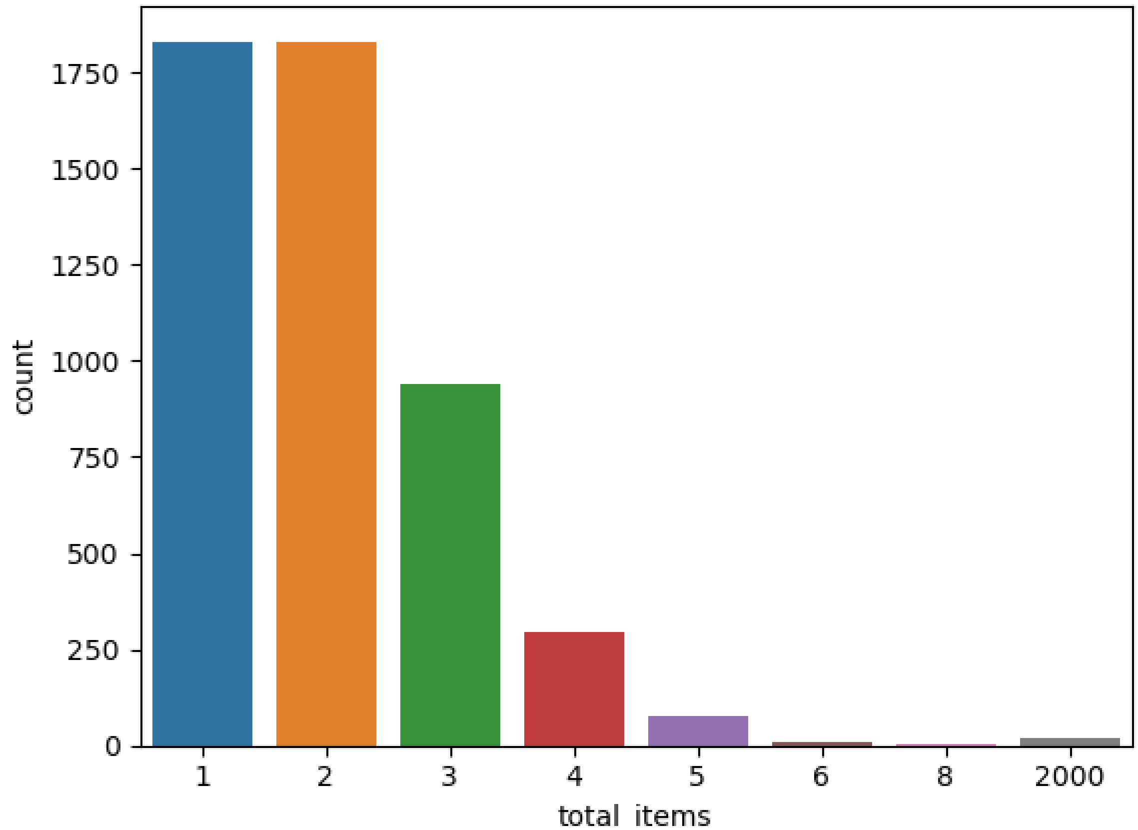
In [9]:
```
sns.boxplot(x=data_frame['order_amount'])
plt.show()
```



In [10]:
```
# We can see that the boxplot is a line on 0, this implies that
# there are many outliers with significant deviation which supports out hypothesis again.
```

In [11]:
```
# Hypothesis 2
```

In [12]:
```
sns.countplot(x="total_items", data= data_frame)
plt.show()
```



In [13]:

```
dict1 = data_frame['total_items'].value_counts().to_dict()
l1 = (list(dict1.items()))
l1.sort()
print (l1)
```

```
[(1, 1830), (2, 1832), (3, 941), (4, 293), (5, 77), (6, 9), (8, 1), (2000, 17)]
```

In [14]:

```
#We see in the graph and dictionary that the frequency of total_items is significantly decraesing.
#Hence, the data points when total_items>5 could be outliers. We will confirm that by also looking at order_amount
```

In [15]:

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
sorted_data_frame = data_frame.sort_values(by=['order_amount'], ascending=False)
print (sorted_data_frame.head(100))
```

```
      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
4868	4869	42	607	704000	2000	credit_card
1362	1363	42	607	704000	2000	credit_card
691	692	78	878	154350	6	debit
2492	2493	78	834	102900	4	debit
3724	3725	78	766	77175	3	credit_card
4420	4421	78	969	77175	3	debit
4192	4193	78	787	77175	3	credit_card
3403	3404	78	928	77175	3	debit
2690	2691	78	962	77175	3	debit
2564	2565	78	915	77175	3	debit
4715	4716	78	818	77175	3	debit
1259	1260	78	775	77175	3	credit_card
2906	2907	78	817	77175	3	debit
3705	3706	78	828	51450	2	credit_card
3101	3102	78	855	51450	2	credit_card
4412	4413	78	756	51450	2	debit
3167	3168	78	927	51450	2	cash
490	491	78	936	51450	2	debit
4079	4080	78	946	51450	2	cash
1529	1530	78	810	51450	2	cash
4311	4312	78	960	51450	2	debit
2818	2819	78	869	51450	2	debit
2821	2822	78	814	51450	2	cash
617	618	78	760	51450	2	cash
2512	2513	78	935	51450	2	debit
511	512	78	967	51450	2	cash
2452	2453	78	709	51450	2	cash
493	494	78	983	51450	2	cash
2495	2496	78	707	51450	2	cash
4040	4041	78	852	25725	1	cash
4918	4919	78	823	25725	1	cash
1056	1057	78	800	25725	1	debit
2922	2923	78	740	25725	1	debit
2270	2271	78	855	25725	1	credit_card
1193	1194	78	944	25725	1	debit
1452	1453	78	812	25725	1	credit_card
3780	3781	78	889	25725	1	cash
4505	4506	78	866	25725	1	debit
2773	2774	78	890	25725	1	cash
3151	3152	78	745	25725	1	credit_card
1384	1385	78	867	25725	1	cash
3085	3086	78	910	25725	1	cash
2548	2549	78	861	25725	1	cash

160	161	78	990	25725	1	credit_card
4584	4585	78	997	25725	1	cash
1419	1420	78	912	25725	1	cash
3440	3441	78	982	25725	1	debit
1204	1205	78	970	25725	1	credit_card
1364	1365	42	797	1760	5	cash
1367	1368	42	926	1408	4	cash
1471	1472	42	907	1408	4	debit
3538	3539	43	830	1086	6	debit
4141	4142	54	733	1064	8	debit
3513	3514	42	726	1056	3	debit
2987	2988	42	819	1056	3	cash
938	939	42	808	1056	3	credit_card
3077	3078	89	754	980	5	debit
2494	2495	50	757	965	5	debit
1563	1564	91	934	960	6	debit
4847	4848	13	993	960	6	cash
2307	2308	61	723	948	6	credit_card
3532	3533	51	828	935	5	cash
1256	1257	6	942	935	5	credit_card
2560	2561	6	845	935	5	credit_card
2039	2040	11	756	920	5	credit_card
3073	3074	90	877	890	5	debit
1150	1151	82	853	885	5	debit
879	880	60	870	885	5	debit
4523	4524	26	995	880	5	credit_card
2032	2033	88	798	880	5	cash
4952	4953	26	786	880	5	cash
1946	1947	33	866	865	5	cash
4958	4959	70	711	865	5	credit_card
2353	2354	27	811	845	5	cash
1962	1963	46	879	830	5	debit
522	523	46	761	830	5	credit_card
2967	2968	46	774	830	5	debit
3865	3866	68	815	816	6	debit
1123	1124	29	911	815	5	credit_card
771	772	19	818	815	5	debit
3927	3928	97	979	810	5	credit_card
2757	2758	66	772	805	5	credit_card
3438	3439	66	842	805	5	credit_card
742	743	12	727	804	4	cash
1764	1765	12	789	804	4	debit

	created_at
2153	2017-03-12 4:00:00
3332	2017-03-24 4:00:00
520	2017-03-02 4:00:00
1602	2017-03-17 4:00:00
60	2017-03-04 4:00:00
2835	2017-03-28 4:00:00
4646	2017-03-02 4:00:00
2297	2017-03-07 4:00:00
1436	2017-03-11 4:00:00
4882	2017-03-25 4:00:00

4056	2017-03-28 4:00:00
15	2017-03-07 4:00:00
1104	2017-03-24 4:00:00
1562	2017-03-19 4:00:00
2969	2017-03-28 4:00:00
4868	2017-03-22 4:00:00
1362	2017-03-15 4:00:00
691	2017-03-27 22:51:43
2492	2017-03-04 4:37:34
3724	2017-03-16 14:13:26
4420	2017-03-09 15:21:35
4192	2017-03-18 9:25:32
3403	2017-03-16 9:45:05
2690	2017-03-22 7:33:25
2564	2017-03-25 1:19:35
4715	2017-03-05 5:10:44
1259	2017-03-27 9:27:20
2906	2017-03-16 3:45:46
3705	2017-03-14 20:43:15
3101	2017-03-21 5:10:34
4412	2017-03-02 4:13:39
3167	2017-03-12 12:23:08
490	2017-03-26 17:08:19
4079	2017-03-20 21:14:00
1529	2017-03-29 7:12:01
4311	2017-03-01 3:02:10
2818	2017-03-17 6:25:51
2821	2017-03-02 17:13:25
617	2017-03-18 11:18:42
2512	2017-03-18 18:57:13
511	2017-03-09 7:23:14
2452	2017-03-27 11:04:04
493	2017-03-16 21:39:35
2495	2017-03-26 4:38:52
4040	2017-03-02 14:31:12
4918	2017-03-15 13:26:46
1056	2017-03-15 10:16:45
2922	2017-03-12 20:10:58
2270	2017-03-14 23:58:22
1193	2017-03-16 16:38:26
1452	2017-03-17 18:09:54
3780	2017-03-11 21:14:50
4505	2017-03-22 22:06:01
2773	2017-03-26 10:36:43
3151	2017-03-18 13:13:07
1384	2017-03-17 16:38:06
3085	2017-03-26 1:59:27
2548	2017-03-17 19:36:00
160	2017-03-12 5:56:57
4584	2017-03-25 21:48:44
1419	2017-03-30 12:23:43
3440	2017-03-19 19:02:54
1204	2017-03-17 22:32:21
1364	2017-03-10 6:28:21

```
1367   2017-03-13 2:38:34
1471  2017-03-12 23:00:22
3538  2017-03-17 19:56:29
4141  2017-03-07 17:05:18
3513  2017-03-24 17:51:05
2987   2017-03-03 9:09:25
938   2017-03-13 23:43:45
3077   2017-03-13 5:27:58
2494   2017-03-04 7:32:45
1563   2017-03-23 8:25:49
4847  2017-03-27 11:00:45
2307  2017-03-26 11:29:37
3532  2017-03-17 16:05:35
1256  2017-03-12 19:49:08
2560  2017-03-16 22:24:30
2039  2017-03-04 10:51:41
3073   2017-03-26 8:08:27
1150  2017-03-24 20:47:47
879   2017-03-27 20:15:11
4523   2017-03-09 8:28:31
2032   2017-03-18 4:24:14
4952   2017-03-17 1:50:18
1946   2017-03-14 5:05:37
4958  2017-03-08 17:22:51
2353   2017-03-13 7:07:39
1962  2017-03-14 17:11:01
522   2017-03-26 19:07:51
2967   2017-03-23 9:22:12
3865   2017-03-11 9:31:50
1123   2017-03-26 0:53:49
771    2017-03-07 8:48:16
3927   2017-03-11 7:37:13
2757   2017-03-14 8:43:29
3438  2017-03-22 17:58:37
742   2017-03-14 16:38:01
1764   2017-03-03 3:10:50
```

In [16]:

```
# We can see here that a lot of bulk purchases are there. They are affecting our calculations.
# Another curious point is that bulk purchses are repeating on the same date and
# have the same shop id, user id, total_items for very differen bulk purchase order amount.
```

In [17]:

```
# A better way to evaulate this data
"""
I have two possible methods
    1. IQR(Inter quartile range) method to remove outliers and then calculate median of remaining data
    2. Calculating the mean of order amounts that lie in the IQR without removing outliers
"""
# Detailed explanations below
```

Out[17]:

```
' \nI have two possible methods\n    1. IQR(Inter quartile range) method to remove outliers and then calculate median of remaining d
```

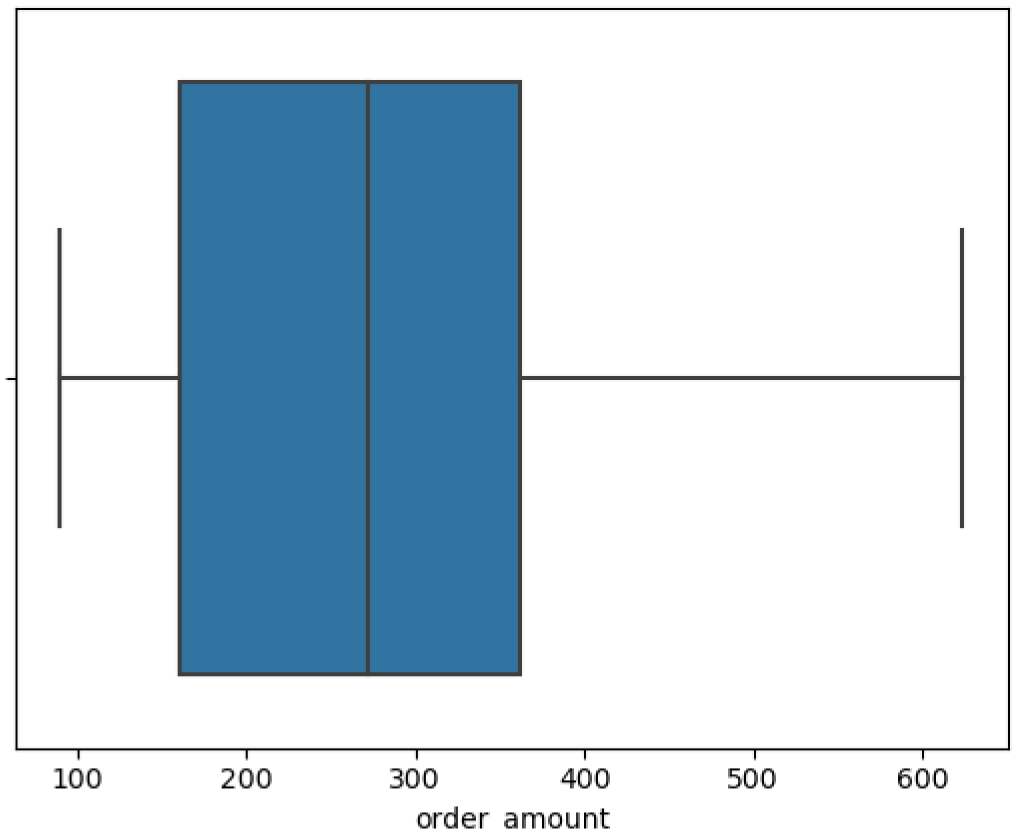
ata\n   2. Calculating the mean of order amounts that lie in the IQR without removing outliers\n'

In [18]:

```
# Method 1 IQR score -> remove outliers -> median
"""
IQR score calculation
Calculate the first and third quartile (Q1 and Q3).
Further, evaluate the interquartile range, IQR = Q3-Q1.
Estimate the lower bound, the lower bound = Q1*1.5 (1.5*IQR rule)
Estimate the upper bound, upper bound = Q3*1.5
Remove the data points that lie outside of the lower and the upper bound.
"""

Q1 = data_frame.order_amount.quantile(0.25)
Q2 = data_frame.order_amount.quantile(0.5)
Q3 = data_frame.order_amount.quantile(0.75)
IQR = Q3 - Q1

# Creates new dataframe without outliers
new_data_frame = data_frame[(data_frame.order_amount < Q2 + IQR * 1.5) & (data_frame.order_amount > Q2 - IQR * 1.5)]
sns.boxplot(x=new_data_frame['order_amount'])
plt.show()
```



In [19]:
```
# We can see that the outliers have been removed from the new data frame.
# Below you can see that the standard deviation has also reduced.
```

In [20]:
```
print (new_data_frame.order_amount.describe())
print ("median = ", new_data_frame.order_amount.median())
```

```
count    4738.000000
mean      283.814268
std       132.061996
min        90.000000
25%       161.000000
50%       272.000000
75%       362.000000
max       624.000000
Name: order_amount, dtype: float64
median =  272.0
```

In [21]:
```
#The AOV here is the median that is equal to $272
```

In [22]:
```
# Method 2: Mean of values in IQR without removing outliers
dict2 = data_frame['order_amount'].value_counts().to_dict()
l2 = (list(dict2.items()))
l2.sort(reverse=True)

sum=0
count=0
for i in range(0, len(l2)):
    if (l2[i][0]<Q3 and l2[i][0]>Q1):
        sum+=l2[i][0]
        count+=1
print ("mean = ", sum/count)
```

```
mean =  275.79761904761904
```

In [23]:
```
#The AOV here is the mean that is equal to $275.79761904761904
```

In [24]:
```
# b. What metric would you report for this dataset?
```

In [25]:
```
"""
The difference between the AOV value in both methods is very less.
But Method 1 is better as it removes the outliers first and then works with the clean data.
"""
```

```
'\nThe difference between the AOV value in both methods is very less. \nBut Method 1 is better as it removes the outliers first and
```

Out[25]: then works with the clean data.\n'

In [26]:
```
# Therefore, the metric I would report is the median of new dataframe
```

In [27]:
```
# c. What is its value?
```

In [28]:
```
# It's value is $272
```