In [1]:
```
#imports
import pandas as pd
import numpy as np
import seaborn as sns
```

Question One

In [2]:
```
#read in the data
data = pd.read_csv('2019 Winter Data Science Intern Challenge Data Set - Sh
data.head()
```

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

In [3]:
```
data.info()
data.isna().sum()
```

```
<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
```

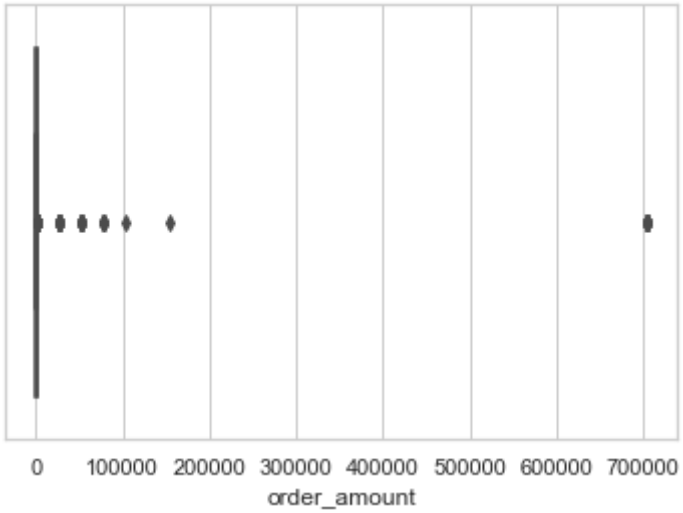
Out[3]:
```
order_id          0
shop_id           0
user_id           0
order_amount      0
total_items       0
payment_method    0
created_at        0
dtype: int64
```

In [4]:
```
#aov = total rev / number of orders
total_rev = sum(data['order_amount'])
total_orders = data.shape[0]
print(total_rev, total_orders)
print('AOC: ' + str(total_rev/ total_orders))
```

```
15725640 5000
AOC: 3145.128
```

In [5]:
```
#boxplot for order amountts
sns.set_theme(style="whitegrid")
sns.boxplot(data = data, x = 'order_amount')
```

Out[5]:
```
<AxesSubplot:xlabel='order_amount'>
```



In [6]:
```
data['order_amount'].sort_values(ascending = False)[:10]
```

Out[6]:
```
2153    704000
3332    704000
520     704000
1602    704000
60      704000
2835    704000
4646    704000
2297    704000
1436    704000
4882    704000
Name: order_amount, dtype: int64
```

In [7]:
```
data.describe()
```

Out[7]:

	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

Part a

It appears that there are some orders that are significantly larger in amount compared to the rest of the dataset. My guess is that there are some rarer shoe sales that cost way more than other shoes. Knowing this, we can attribute the absurdly high average order value to this irregularity. A better way to evaluate the data and better gauge consumer tendencies would be to use the median. Alternatively, we can remove the outliers and recalculate the mean.

Part b

An alternative metric to use would be to use the median.

Part c

In [8]:
```
print('Median of order_amounts: ' + str(np.median(data['order_amount'])))
```

```
Median of order_amounts: 284.0
```

Question 2

Part a

Query:

select count(*)

```
from Orders o, Shippers s

where s.ShipperName = 'Speedy Express' and s.ShipperID = o.ShipperID
```

Answer:

54

Part b

Query:

```
select e.LastName from Employees e

join Orders o on e.EmployeeID = o.EmployeeID

group by o.EmployeeID

order by count(o.EmployeeID) desc limit 1
```

Answer:

Peacock

Part c

Query:

```
select ProductName from

OrderDetails od

join Orders o on o.OrderID = od.OrderID

join Customers c on c.CustomerID = o.CustomerID

join Products p on p.ProductID = od.ProductID

where Country = "Germany"

group by ProductName

order by count(ProductName) desc

limit 1
```

Answer:

Gorgonzola Telino

In [ ]: