In [1]:
```
import pandas as pd
import numpy as np

data = pd.read_csv("2019 Winter Data Science Intern Challenge Data Set.csv")
```

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)](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.

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

In [2]:
```
# Create a derived column for the price_per_item
data['price_per_item'] = data['order_amount']/data['total_items']
```

The price_per_item is an important metric to consider when evaluating outliers. This metric will be used later in the Exploratory Data Analysis (EDA).

In [3]:
```
# Create a simple summary statistics table for the dataset
data.describe()
```

Out[3]:

	order_id	shop_id	user_id	order_amount	total_items	price_per_item
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000	5000.000000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720	387.742800
std	1443.520003	29.006118	87.798982	41282.539349	116.32032	2441.963725
min	1.000000	1.000000	607.000000	90.000000	1.00000	90.000000
25%	1250.750000	24.000000	775.000000	163.000000	1.00000	133.000000
50%	2500.500000	50.000000	849.000000	284.000000	2.00000	153.000000
75%	3750.250000	75.000000	925.000000	390.000000	3.00000	169.000000
max	5000.000000	100.000000	999.000000	704000.000000	2000.00000	25725.000000

In [4]:
```
# Create a simple histogram of the distribution of order_amount
hist = data.order_amount.hist()
```

The summary statistics table shows that the abnormally large AOV of $3145.13 is caused by extremely large outlier which bias the order_amount mean. This is evidenced by the order_amount quartiles. The 75th quartile is \$390, while the maximum order_amount in the dataset is $704,000. The histogram shows that there are extreme values over \$100,000, lending further evidence that outliers are biasing the AOV calculation.

The minimum order_amount is $90, which seems to be a reasonable size for an order of shoes. Therefore, all the outliers must be at the high end of the range of data.

In [5]:
```
# List the unique order_amount greater than $1000
print('Unique order_amount values greater than $1000:\n' +
      str(np.unique(data[data['order_amount'] > 1000].order_amount)))
```

```
Unique order_amount values greater than $1000:
[  1056   1064   1086   1408   1760  25725  51450  77175 102900 154350
 704000]
```

By listing the unique order amounts that are greater than $1000, we can see there are six outlier amounts (shown by the significant jump in order_amount from \$1760 to $25725.

In [6]:
```
# Count the number of outlier values (order_amount greater than $25,000)
print('Count of order_amount values greater than $25,000: ' +
      str(data[data['order_amount'] > 25000].shape[0]))
```

```
Count of order_amount values greater than $25,000: 63
```

In [7]:
```
# List the unique shop_ids that have outlier order_amount values
print('Unique shops with outlier order_amount values: '+
      str(np.unique(data[data['order_amount'] > 25000].shop_id)))
```

```
Unique shops with outlier order_amount values: [42 78]
```

We see that there are 63 total outlier observations coming from 2 different shops. We now turn to examine each of the shops.

In [8]:
```
# Display all the orders from shop_id 48
display(
    data[data.shop_id == 42].sort_values(
        by = 'order_amount', ascending = False))
```

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	price_
15	16	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	
4868	4869	42	607	704000	2000	credit_card	2017-03-22 4:00:00	
4646	4647	42	607	704000	2000	credit_card	2017-03-02 4:00:00	
4056	4057	42	607	704000	2000	credit_card	2017-03-28 4:00:00	
3332	3333	42	607	704000	2000	credit_card	2017-03-24 4:00:00	
2969	2970	42	607	704000	2000	credit_card	2017-03-28 4:00:00	
2835	2836	42	607	704000	2000	credit_card	2017-03-28 4:00:00	
2297	2298	42	607	704000	2000	credit_card	2017-03-07 4:00:00	
1602	1603	42	607	704000	2000	credit_card	2017-03-17 4:00:00	
1562	1563	42	607	704000	2000	credit_card	2017-03-19 4:00:00	
2153	2154	42	607	704000	2000	credit_card	2017-03-12 4:00:00	
4882	4883	42	607	704000	2000	credit_card	2017-03-25 4:00:00	
1104	1105	42	607	704000	2000	credit_card	2017-03-24 4:00:00	
1362	1363	42	607	704000	2000	credit_card	2017-03-15 4:00:00	
60	61	42	607	704000	2000	credit_card	2017-03-04 4:00:00	
520	521	42	607	704000	2000	credit_card	2017-03-02 4:00:00	
1364	1365	42	797	1760	5	cash	2017-03-10 6:28:21	
1471	1472	42	907	1408	4	debit	2017-03-12 23:00:22	
1367	1368	42	926	1408	4	cash	2017-03-13 2:38:34	
3513	3514	42	726	1056	3	debit	2017-03-24 17:51:05	
938	939	42	808	1056	3	credit_card	2017-03-13 23:43:45	

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	price_
2987	2988	42	819	1056	3	cash	2017-03-03 9:09:25	
1520	1521	42	756	704	2	debit	2017-03-22 13:10:31	
2766	2767	42	970	704	2	credit_card	2017-03-05 10:45:42	
4767	4768	42	720	704	2	credit_card	2017-03-14 10:26:08	
4421	4422	42	736	704	2	credit_card	2017-03-01 12:19:49	
4326	4327	42	788	704	2	debit	2017-03-16 23:37:57	
4294	4295	42	859	704	2	cash	2017-03-24 20:50:40	
409	410	42	904	704	2	credit_card	2017-03-04 14:32:58	
835	836	42	819	704	2	cash	2017-03-09 14:15:15	
2609	2610	42	868	704	2	debit	2017-03-23 18:10:14	
2491	2492	42	868	704	2	debit	2017-03-01 18:33:33	
1911	1912	42	739	704	2	cash	2017-03-07 5:42:52	
2273	2274	42	747	704	2	debit	2017-03-27 20:48:19	
2003	2004	42	934	704	2	cash	2017-03-26 9:21:26	
2053	2054	42	951	352	1	debit	2017-03-19 11:49:12	
1929	1930	42	770	352	1	credit_card	2017-03-17 8:11:13	
4745	4746	42	872	352	1	debit	2017-03-24 0:57:24	
308	309	42	770	352	1	credit_card	2017-03-11 18:14:39	
4625	4626	42	809	352	1	credit_card	2017-03-11 8:21:26	

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	price_
2018	2019	42	739	352	1	debit	2017-03-01 12:42:26	
4231	4232	42	962	352	1	cash	2017-03-04 0:01:19	
834	835	42	792	352	1	cash	2017-03-25 21:31:25	
40	41	42	793	352	1	credit_card	2017-03-24 14:15:41	
3998	3999	42	886	352	1	debit	2017-03-09 20:10:41	
3903	3904	42	975	352	1	debit	2017-03-12 1:28:31	
3697	3698	42	839	352	1	debit	2017-03-12 2:45:09	
3651	3652	42	830	352	1	credit_card	2017-03-24 22:26:58	
979	980	42	744	352	1	debit	2017-03-12 13:09:04	
1512	1513	42	946	352	1	debit	2017-03-24 13:35:04	

We can see from above that shop_id 42 sells a sneaker for $352 each. The outlier order_amount values of \$704,000 result from bulk orders of 2,000 sneakers. Returning to our simple summary statistics table from above, these bulk orders are the largest (maximum) total_items in our dataset. It seems reasonable to trim these bulk orders from our dataset before calculating the mean.

In [9]:
```
# Display all the orders from shop_id 78
display(
    data[data.shop_id == 78].sort_values(
        by = 'order_amount', ascending = False))
```

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	price_
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	
1259	1260	78	775	77175	3	credit_card	2017-03-27 9:27:20	
3724	3725	78	766	77175	3	credit_card	2017-03-16 14:13:26	
2906	2907	78	817	77175	3	debit	2017-03-16 3:45:46	
2690	2691	78	962	77175	3	debit	2017-03-22 7:33:25	
2564	2565	78	915	77175	3	debit	2017-03-25 1:19:35	
4192	4193	78	787	77175	3	credit_card	2017-03-18 9:25:32	
4420	4421	78	969	77175	3	debit	2017-03-09 15:21:35	
3403	3404	78	928	77175	3	debit	2017-03-16 9:45:05	
4715	4716	78	818	77175	3	debit	2017-03-05 5:10:44	
617	618	78	760	51450	2	cash	2017-03-18 11:18:42	
3167	3168	78	927	51450	2	cash	2017-03-12 12:23:08	
3705	3706	78	828	51450	2	credit_card	2017-03-14 20:43:15	
3101	3102	78	855	51450	2	credit_card	2017-03-21 5:10:34	
493	494	78	983	51450	2	cash	2017-03-16 21:39:35	
511	512	78	967	51450	2	cash	2017-03-09 7:23:14	
2821	2822	78	814	51450	2	cash	2017-03-02 17:13:25	
490	491	78	936	51450	2	debit	2017-03-26 17:08:19	
4412	4413	78	756	51450	2	debit	2017-03-02 4:13:39	

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	price_
2818	2819	78	869	51450	2	debit	2017-03-17 6:25:51	
4079	4080	78	946	51450	2	cash	2017-03-20 21:14:00	
2512	2513	78	935	51450	2	debit	2017-03-18 18:57:13	
2495	2496	78	707	51450	2	cash	2017-03-26 4:38:52	
2452	2453	78	709	51450	2	cash	2017-03-27 11:04:04	
1529	1530	78	810	51450	2	cash	2017-03-29 7:12:01	
4311	4312	78	960	51450	2	debit	2017-03-01 3:02:10	
4040	4041	78	852	25725	1	cash	2017-03-02 14:31:12	
3440	3441	78	982	25725	1	debit	2017-03-19 19:02:54	
3780	3781	78	889	25725	1	cash	2017-03-11 21:14:50	
4584	4585	78	997	25725	1	cash	2017-03-25 21:48:44	
4505	4506	78	866	25725	1	debit	2017-03-22 22:06:01	
160	161	78	990	25725	1	credit_card	2017-03-12 5:56:57	
3151	3152	78	745	25725	1	credit_card	2017-03-18 13:13:07	
3085	3086	78	910	25725	1	cash	2017-03-26 1:59:27	
2922	2923	78	740	25725	1	debit	2017-03-12 20:10:58	
2773	2774	78	890	25725	1	cash	2017-03-26 10:36:43	
2548	2549	78	861	25725	1	cash	2017-03-17 19:36:00	
2270	2271	78	855	25725	1	credit_card	2017-03-14 23:58:22	

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at	price_
1452	1453	78	812	25725	1	credit_card	2017-03-17 18:09:54	
1419	1420	78	912	25725	1	cash	2017-03-30 12:23:43	
1384	1385	78	867	25725	1	cash	2017-03-17 16:38:06	
1204	1205	78	970	25725	1	credit_card	2017-03-17 22:32:21	
1193	1194	78	944	25725	1	debit	2017-03-16 16:38:26	
1056	1057	78	800	25725	1	debit	2017-03-15 10:16:45	
4918	4919	78	823	25725	1	cash	2017-03-15 13:26:46	

We can see from above that shop_id 78 sells a sneaker for $25,725 each, causing all of its order_amount observations to be outliers, since the price is unusually high for a sneaker when compared to the rest of our dataset. Returning to our simple summary statistics table from above, these sneakers are the largest (maximum) price_per_item in our dataset. It seems reasonable to trim all of the observations from shop_id 78 from our calculation of the mean since its sneaker is so much more expensive than the rest of the shops.

In [10]:
```
# Trim all observations with total_items equal to 2000
data = data.drop(data[data['total_items'] == 2000].index)
# Trim all observations from shop_id 78
data = data.drop(data[data['shop_id'] == 78].index)

# Create a simple summary statistics table for the trimmed dataset
data.describe()
```

Out[10]:

	order_id	shop_id	user_id	order_amount	total_items	price_per_item
count	4937.000000	4937.000000	4937.000000	4937.000000	4937.000000	4937.000000
mean	2499.551347	49.846465	849.752279	302.580514	1.994734	151.788536
std	1444.069407	29.061131	86.840313	160.804912	0.982821	29.034215
min	1.000000	1.000000	700.000000	90.000000	1.000000	90.000000
25%	1248.000000	24.000000	775.000000	163.000000	1.000000	132.000000
50%	2497.000000	50.000000	850.000000	284.000000	2.000000	153.000000
75%	3751.000000	74.000000	925.000000	387.000000	3.000000	166.000000
max	5000.000000	100.000000	999.000000	1760.000000	8.000000	352.000000

At first glance, the summary statistics table for the trimmed dataset seems to have much more reasonable values than the full dataset, particularly when looking at the min, max, and quartile values for order_amount, total_items, and price_per_item.

In [11]:
```
# Calculating the new "trimmed" AOV with 63 outliers removed
print(f'Trimmed AOV: $', round(np.mean(data.order_amount), 2))
```

```
Trimmed AOV: $ 302.58
```

Summarized Answer to Question 1

1. What is going wrong with the naive AOV calculation of $3145.13? What could be a better way to evaluate this data?

    - *The naive AOV calculation was heavily biased upwards by the presence of 63 extremely large outlier observations. These outlier observations come from two shops and fall into one of two categories.*
    - *If the outlier observation is from shop_id 42, then it is a result of bulk orders of 2000 items at once, causing the order_amount to be $704,000.*
    - *If the outlier observation is from shop_id 78, then it is a result of extremely expensive sneakers. Each sneaker from this shop costs $25,725, causing all the order_amount observations from this shop to be extremely large outlier observations.*
    - *The best method to evaluate this data is to trim the 63 outliers out of the dataset and then recalculate the AOV. I verified this method by comparing the summary statistics table before and after the outlier observations were trimmed to ensure that the distribution of observations appeared to be reasonable for order_amount, total_items, and price_per_item (a column derived from the other two columns).*

1. What metric would you report for this datset?

    - *I would report the trimmed AOV that results from trimming the 63 outliers out of our dataset.*

1. What is its value?

    - *The trimmed AOV value is* **$302.58.**

Question 2

1. How many orders were shipped by Speedy Express in total?

    - *Using the query below, I found that* ***54*** *orders were shipped by Speedy Express in total.*

```
SELECT Count() AS 'Number of Speedy Express Orders'

FROM Shippers, Orders

WHERE Shippers.ShipperName = 'Speedy Express' AND Shippers.ShipperID = Orders.ShipperID;
```

1. What is the last name of the employees with the most order?

    - *Using the query below, I found that* ***Peacock*** *is the last name of the employee with the most orders (40 orders in total).*

SELECT MAX(x.num) AS 'Number of Orders', x.LastName AS 'Last Name of Employee with the most orders'

FROM(Select Count() as num, Orders.EmployeeID, Employees.LastName

```
FROM Orders

LEFT JOIN Employees

ON Orders.EmployeeID=Employees.EmployeeID

GROUP BY Orders.EmployeeID) x
```

1. What product was ordered the most by customers in Germany?
    - *Using the query below, I found that **Boston Crab Meat** with ProductID **40** is the product ordered the most by customers in Germany.*

SELECT ProductName AS 'Name of Product that was most ordered by customers in Germany', ProductID AS 'ID of Product', MAX(x.num) AS 'Quantity of Product ordered by German customers'

FROM(SELECT Products.ProductID, Products.ProductName, SUM(Quantity) AS num

```
FROM OrderDetails

JOIN Orders

ON OrderDetails.OrderID = Orders.OrderID

JOIN Customers

ON Orders.CustomerID = Customers.CustomerID

JOIN Products

ON OrderDetails.ProductID = Products.ProductID

WHERE Customers.Country = 'Germany'

Group By OrderDetails.ProductID) x
```

In [ ]: