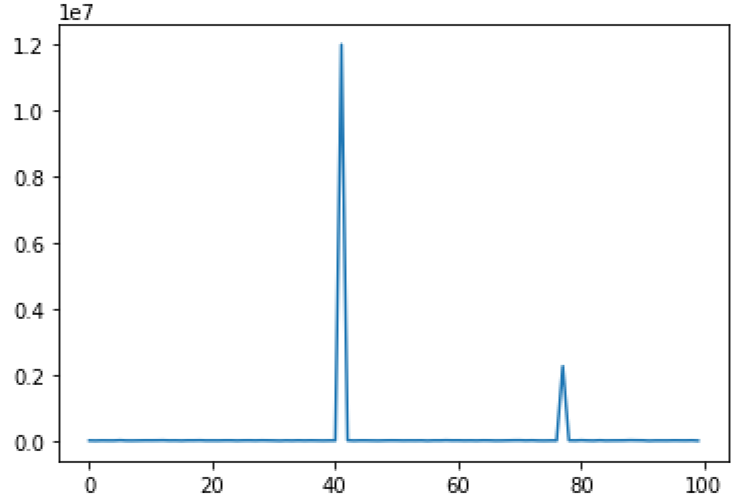
Winter 2021 Data Science Intern Challenge

Please complete the following questions, and provide your thought process/work. You can attach your work in a text file, link, etc. on the application page. Please ensure answers are easily visible for reviewers!

Question 1: Given some sample data, write a program to answer the following: click here to access the required data set

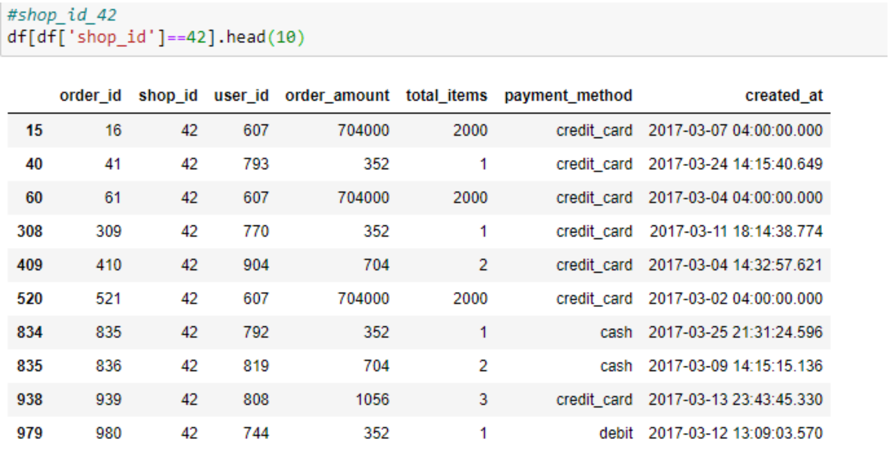
On Shopify, we have exactly 100 sneaker shops, and each of these shops sells only one model of shoe. We want to do some analysis of the average order value (AOV). When we look at orders data over a 30 day window, we naively calculate an AOV of $3145.13. Given that we know these shops are selling sneakers, 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.



This is the graph for Order amount and shopID, from that we can clearly see two spikes one at 42 and another at 78. The unconvincing AOV price should come from these two unusual shops.

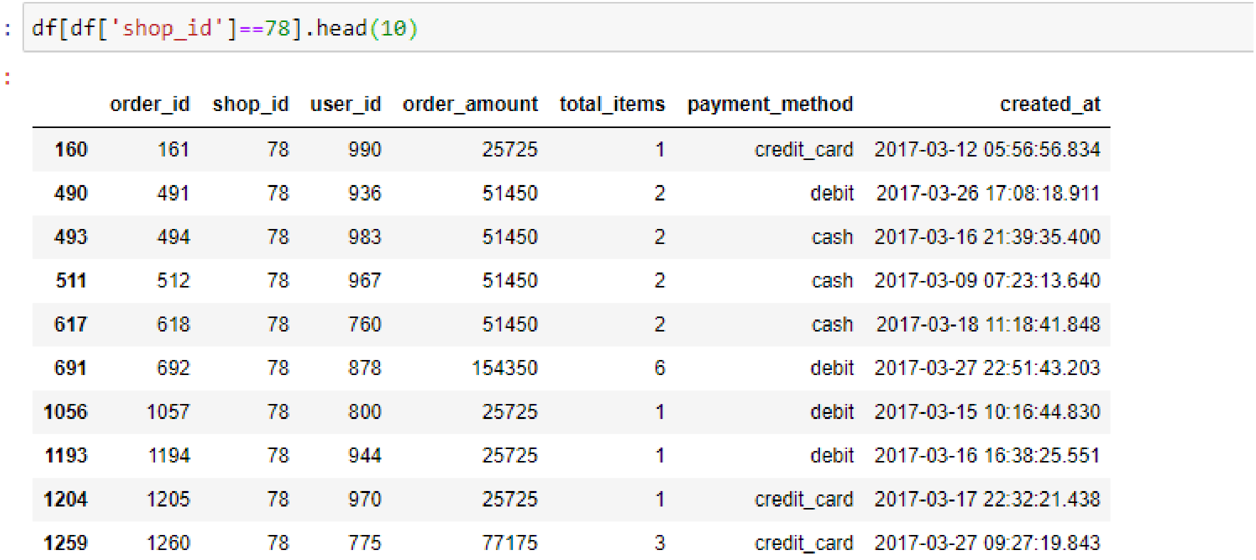
After looking into these two shops:



```
#shop_id_42
df[df['shop_id']==42].head(10)
```

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
15	16	42	607	704000	2000	credit_card	2017-03-07 04:00:00.000
40	41	42	793	352	1	credit_card	2017-03-24 14:15:40.649
60	61	42	607	704000	2000	credit_card	2017-03-04 04:00:00.000
308	309	42	770	352	1	credit_card	2017-03-11 18:14:38.774
409	410	42	904	704	2	credit_card	2017-03-04 14:32:57.621
520	521	42	607	704000	2000	credit_card	2017-03-02 04:00:00.000
834	835	42	792	352	1	cash	2017-03-25 21:31:24.596
835	836	42	819	704	2	cash	2017-03-09 14:15:15.136
938	939	42	808	1056	3	credit_card	2017-03-13 23:43:45.330
979	980	42	744	352	1	debit	2017-03-12 13:09:03.570

We can found the price for one pair is not high but there are some 2000 pairs orders in shop 42 everyday at 4. It seems like the transactions are probably some sort of supplier purchasing many shoes at once since the order amount is consistently 2000.



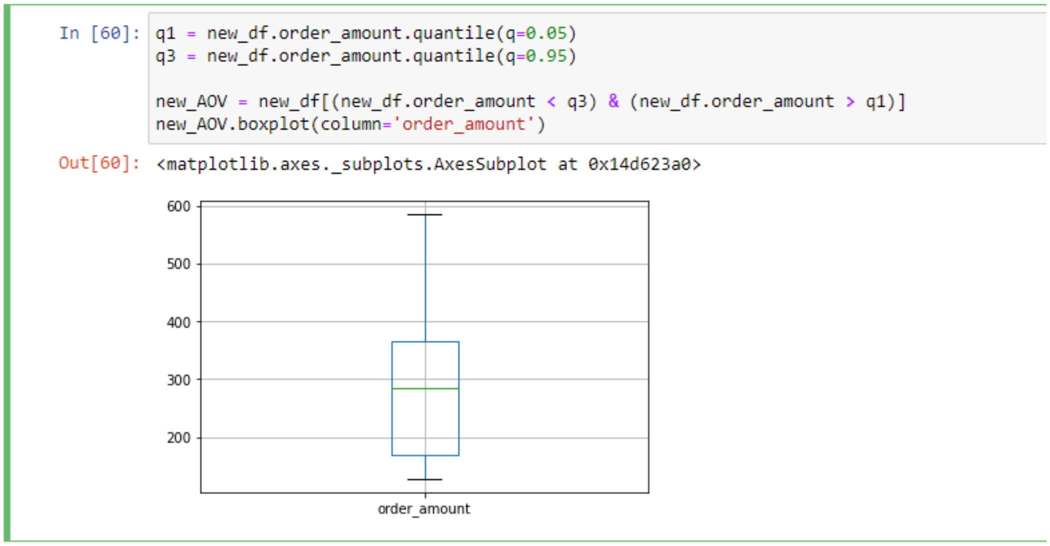
```
: df[df['shop_id']==78].head(10)
```

:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
160	161	78	990	25725	1	credit_card	2017-03-12 05:56:56.834
490	491	78	936	51450	2	debit	2017-03-26 17:08:18.911
493	494	78	983	51450	2	cash	2017-03-16 21:39:35.400
511	512	78	967	51450	2	cash	2017-03-09 07:23:13.640
617	618	78	760	51450	2	cash	2017-03-18 11:18:41.848
691	692	78	878	154350	6	debit	2017-03-27 22:51:43.203
1056	1057	78	800	25725	1	debit	2017-03-15 10:16:44.830
1193	1194	78	944	25725	1	debit	2017-03-16 16:38:25.551
1204	1205	78	970	25725	1	credit_card	2017-03-17 22:32:21.438
1259	1260	78	775	77175	3	credit_card	2017-03-27 09:27:19.843

For shop 78, it’s high total order amount comes from the high price of each shoes pair (25725 for each). It’s possible that shop 78 sells a kind of luxury sneakers which has higher order amount than usual ones.

Therefore, to get a more reliable AOV, I would firstly remove orders from these two shops as they are outliers. Then I used boxplot to find the median of rest data in range (5%-95%)

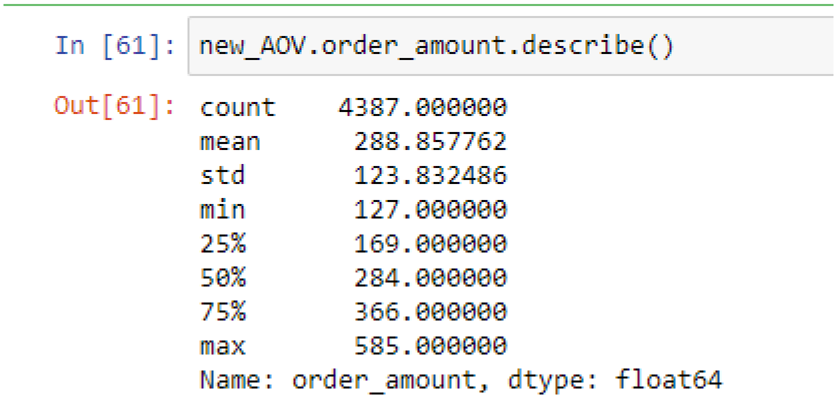


```
In [60]: q1 = new_df.order_amount.quantile(q=0.05)
         q3 = new_df.order_amount.quantile(q=0.95)

         new_AOV = new_df[(new_df.order_amount < q3) & (new_df.order_amount > q1)]
         new_AOV.boxplot(column='order_amount')

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x14d623a0>
```

2. What metric would you report for this dataset?
   I would use the median values of the data in the new boxplot which remove the first and last 5% of data. After removing all outliers, the median can better represent the AOV price of most shoes. Also, the std decreased a lot after removing outliers which also tells that this is a better approach to calculate AOV.
3. What is its value?
   As the result in he table below, the data I got is 284 which is a more convincing number.



```
In [61]: new_AOV.order_amount.describe()

Out[61]: count    4387.000000
         mean      288.857762
         std       123.832486
         min       127.000000
         25%       169.000000
         50%       284.000000
         75%       366.000000
         max       585.000000
         Name: order_amount, dtype: float64
```

Question 2: For this question you’ll need to use SQL. Follow this link to access the data set required for the challenge. Please use queries to answer the following questions. Paste your queries along with your final numerical answers below.

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

The output shows the number of orders is **54**.

```
SELECT COUNT(*) AS SpeedyExpressNum

FROM Orders

JOIN Shippers

  ON Shippers.ShipperID = Orders.ShipperID

WHERE Shippers.ShipperName = 'Speedy Express'
```

2. What is the last name of the employee with the most orders?

The employee with the last name **Peacock** had the most orders at **40**

```
SELECT LastName,most_freq_order AS NumberOfOrders

FROM (

        (SELECT EmployeeID, MAX(sales) AS most_freq_order

        FROM (

                SELECT  EmployeeID, Count(OrderID) AS sales

                FROM    Orders

                GROUP BY  EmployeeID

                )

         ) AS t1

        JOIN

         (SELECT  EmployeeID, LastName

            FROM    Employees

          ) AS t2

        ON t1.EmployeeID == t2.EmployeeID

)
```

3. What product was ordered the most by customers in Germany?

Boston Crab Meat has the most orders at **160 total orders**.

```
SELECT ProductName,Most_Freq.most_freq_order As TotalOrder

FROM Products

JOIN

(SELECT ProductID, MAX(TotalQantity) AS most_freq_order

        FROM

                (SELECT ProductID,SUM(Quantity) as TotalQantity

                FROM OrderDetails

         JOIN

                (SELECT Orders.OrderID, Customers.Country

                FROM Orders

                JOIN Customers

                    ON Customers.CustomerID = Orders.CustomerID

                WHERE Customers.Country = 'Germany') AS t1

        ON OrderDetails.OrderID == t1.OrderID

        GROUP BY  ProductID

 )) AS Most_Freq

ON Products.ProductID = Most_Freq.ProductID
```