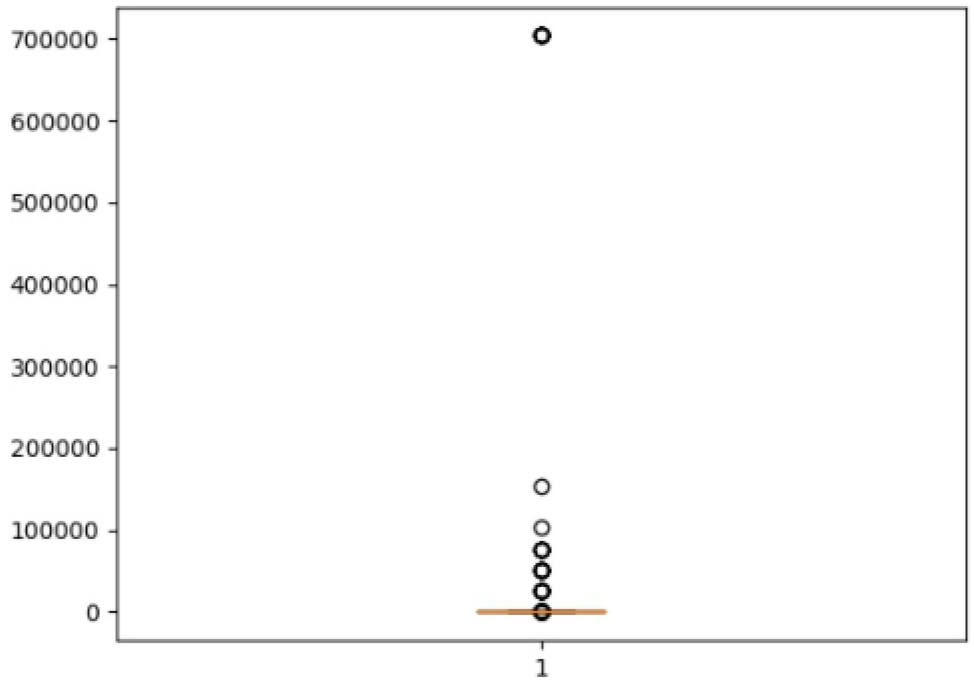
Diwei Guan

Shopify Data Science Challenge Winter 2022

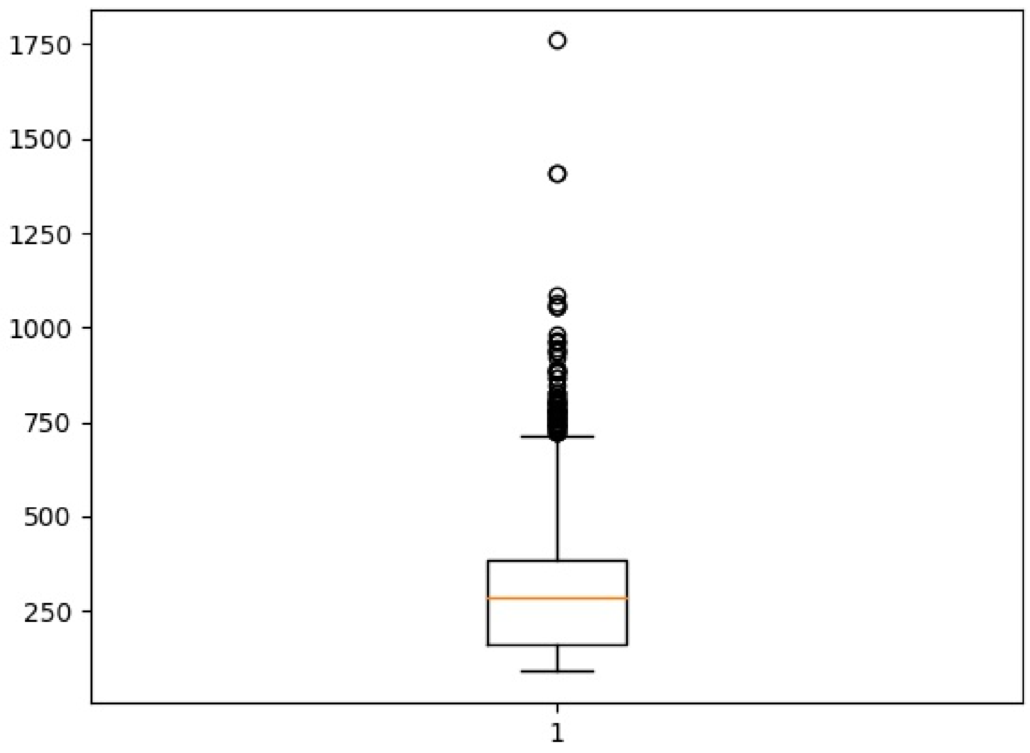
Part A

1.) It is naively done because they do the sum of the order_amount / total # of orders, but it seems that some orders are driving up the average value, so below we go through a strategy to remove such values. Even looking through the data manually we can see some values that are way over the top and can definitely be considered outliers. We are also assuming that the data given has been cleaned, which means that there are no duplicates and null values, which might cause our calculation to be wrong.

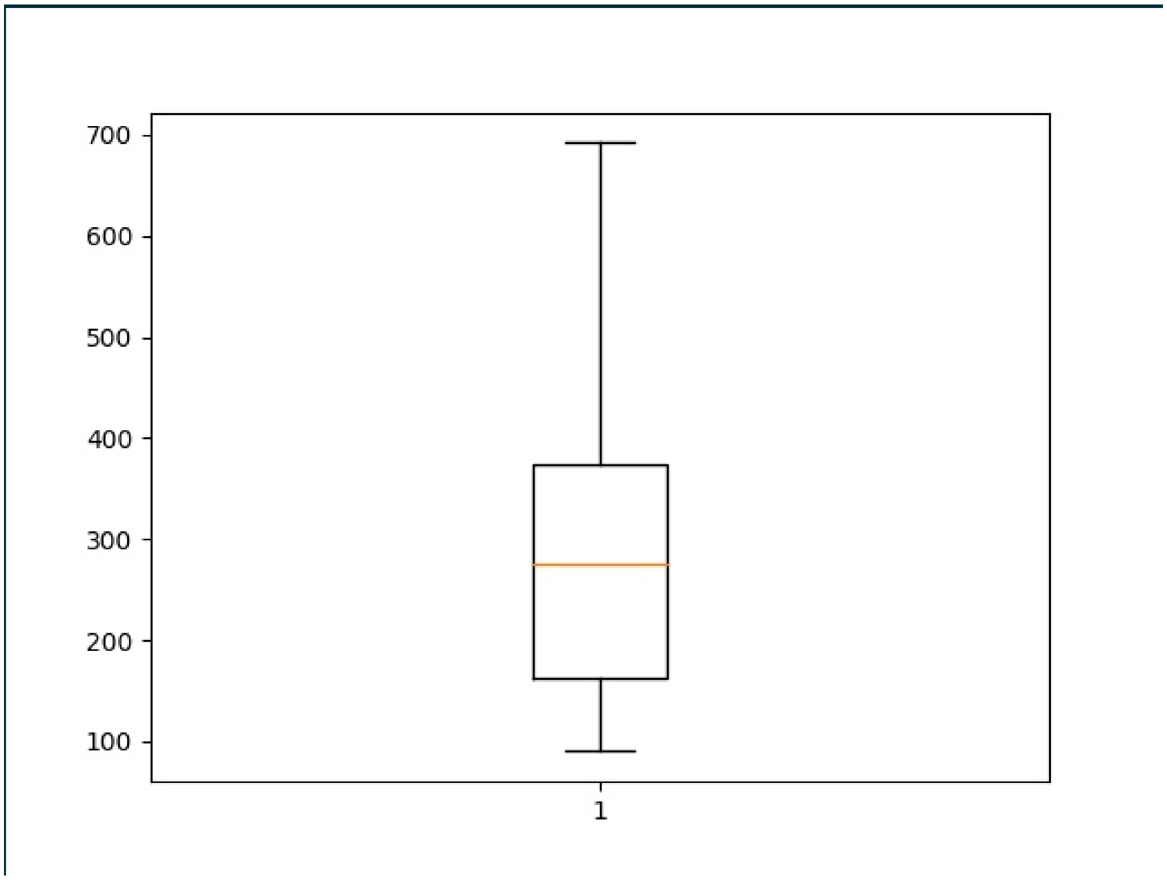
2.) Instead you can use the same metric that is used in the original calculation but I would remove the outliers in the dataset. Using the box plot given we can see a lot of the outliers lie among the > 10000 order amounts because the box plot below barely has a visible box,



these values inflate the average given, therefore I would remove them and re-do the calculation. After removing everything > 10000, we still have some outliers as seen in the box plot below:



After removing everything > 700, we can see the box plot now no longer has any outliers. So we use this range of values to calculate the AOV (which is talked about below.)



A completely separate metric we could consider looking at is the average shoe value, which would be to look at sum of order_amount / # of shoes purchased.

3.) The AOV is $302.58 after removing the initial outlier data, which is a much more reasonable number than the previous calculated value of $3145.13. After removing every order_amount over 10000, I took a look at the box plot again and noticed there are still some values that are over the range mainly those greater than 700, so I adjusted the box plot range and it can be seen in the third picture shown in question 2, that I have a good range now. So then I calculate the AOV after that and the output is $288.87. The average shoe value is also something we could consider, the average shoe value is $357.92.

Part B

1.) 54,
Query used:
```
SELECT COUNT(*) FROM [Orders]
LEFT JOIN [Shippers]
ON [Orders].ShipperID = [Shippers].ShipperID
WHERE LOWER(ShipperName) LIKE '%speedy express%'
```

2.) Peacock,
Query used:

```
SELECT Orders.EmployeeID, COUNT(*) AS NUM_OF_ORDERS, LastName
FROM [Orders] AS Orders
LEFT JOIN [Employees] AS Employees
ON Orders.EmployeeID = Employees.EmployeeID
GROUP BY Orders.EmployeeID
ORDER BY NUM_OF_ORDERS DESC
```

3.) Boston Crab Meat
Query used:

```
SELECT
Country,
PRODUCTS.ProductID,
SUM(Quantity) AS TOTAL_QUANTITY,
ProductName
FROM [Orders] AS ORDERS
LEFT JOIN [Customers] AS CUSTOMERS
ON ORDERS.CustomerID = CUSTOMERS.CustomerID
LEFT JOIN [OrderDetails] AS DETAILS
ON ORDERS.OrderID = DETAILS.OrderID
LEFT JOIN [Products] AS PRODUCTS
ON DETAILS.ProductID = PRODUCTS.ProductID
WHERE LOWER(CUSTOMERS.Country) LIKE '%germany%'
GROUP BY COUNTRY, PRODUCTS.ProductID, ProductName
ORDER BY TOTAL_QUANTITY DESC
```