Shopify Fall 2021 Data Challenge

1(a)

To begin, let's make sure we can recreate the naive calculation of the AOV:

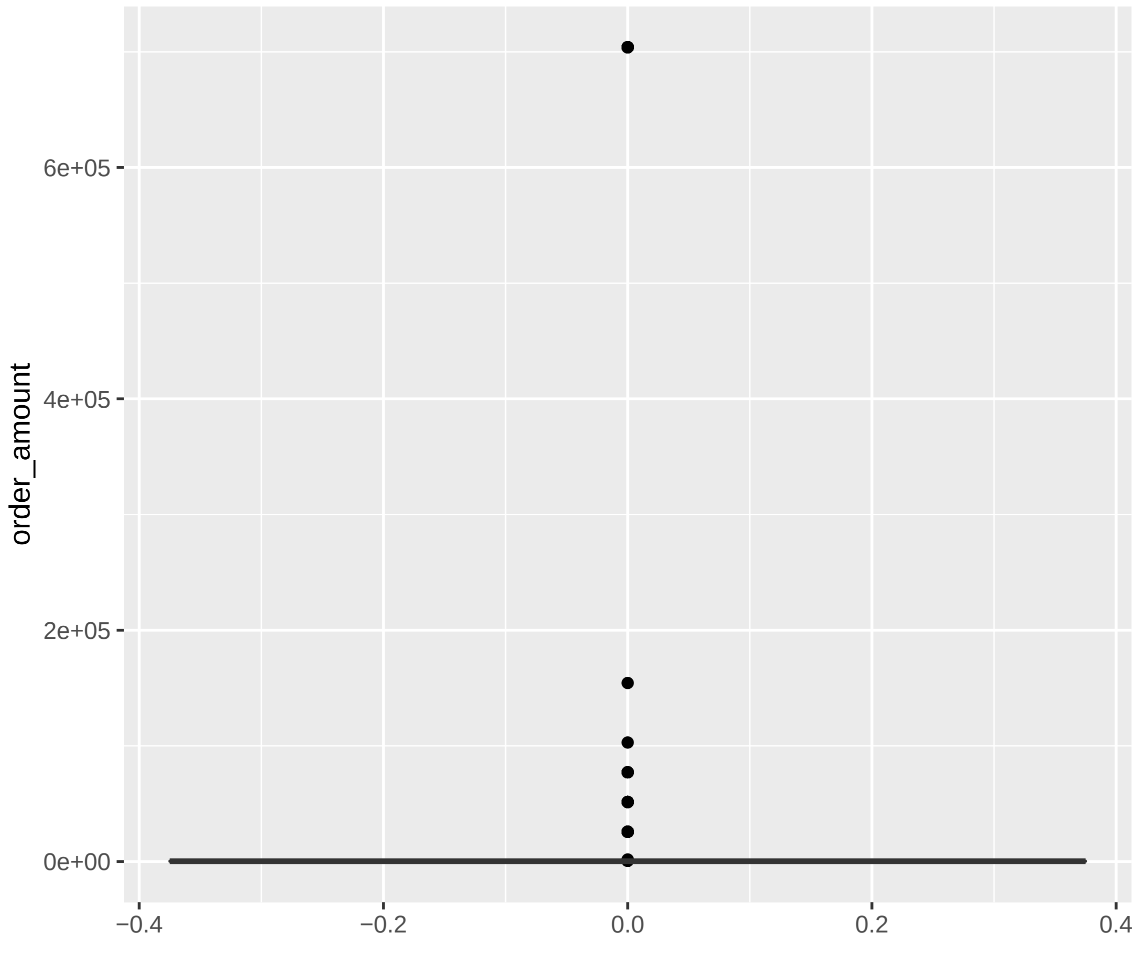
```
library(tidyverse)
data <- read_csv("2019 Winter Data Science Intern Challenge Data Set - Sheet1.csv")

AOV <- mean(data$order_amount)
AOV
```

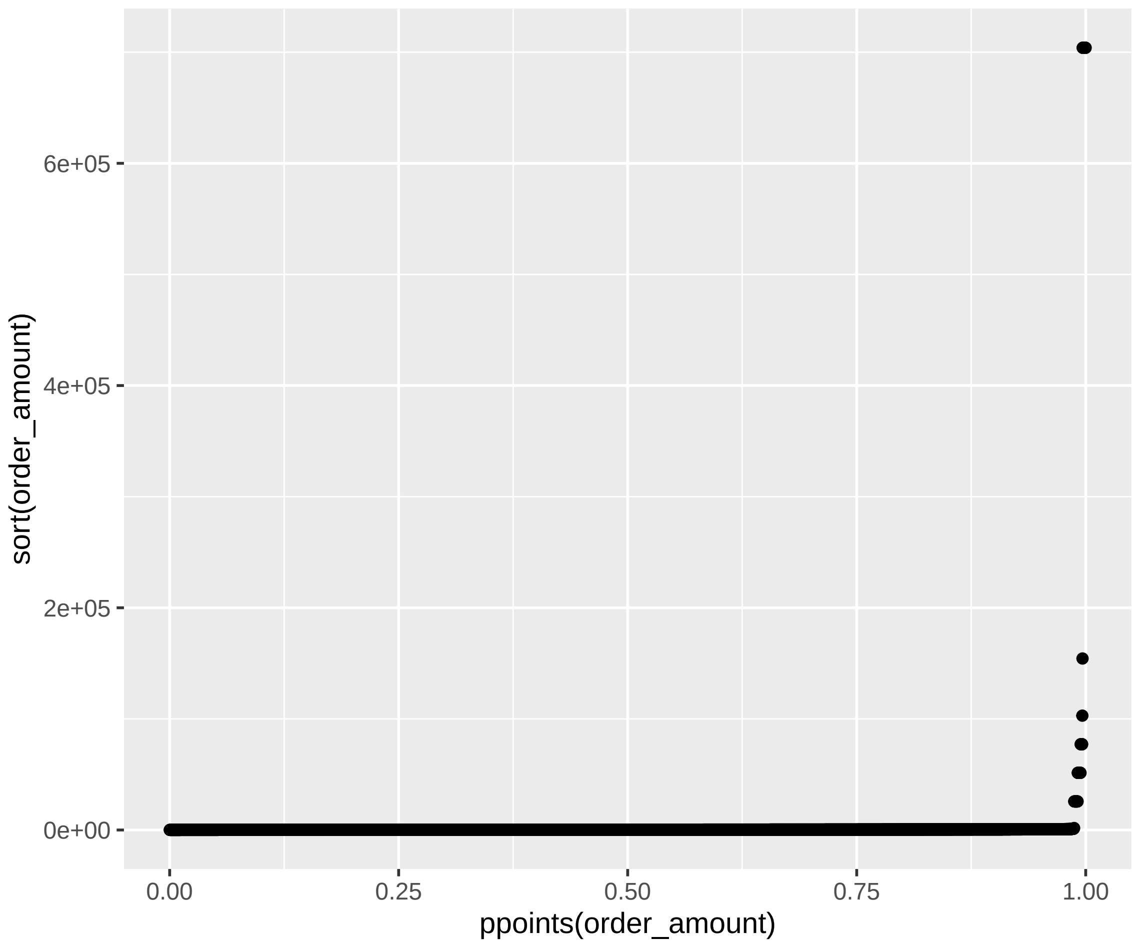
```
## [1] 3145.128
```

Now that we have made sure we can recreate the flawed value, it could be useful to get a feel for the data with some initial visual analysis. I want to get an idea of how the order amounts are distributed, so let's take a look at a box plot and quantile plot of the order amounts for all orders within the data:

```
data %>% ggplot(mapping = aes(y=order_amount)) +
         geom_boxplot(outlier.color = "black")
```

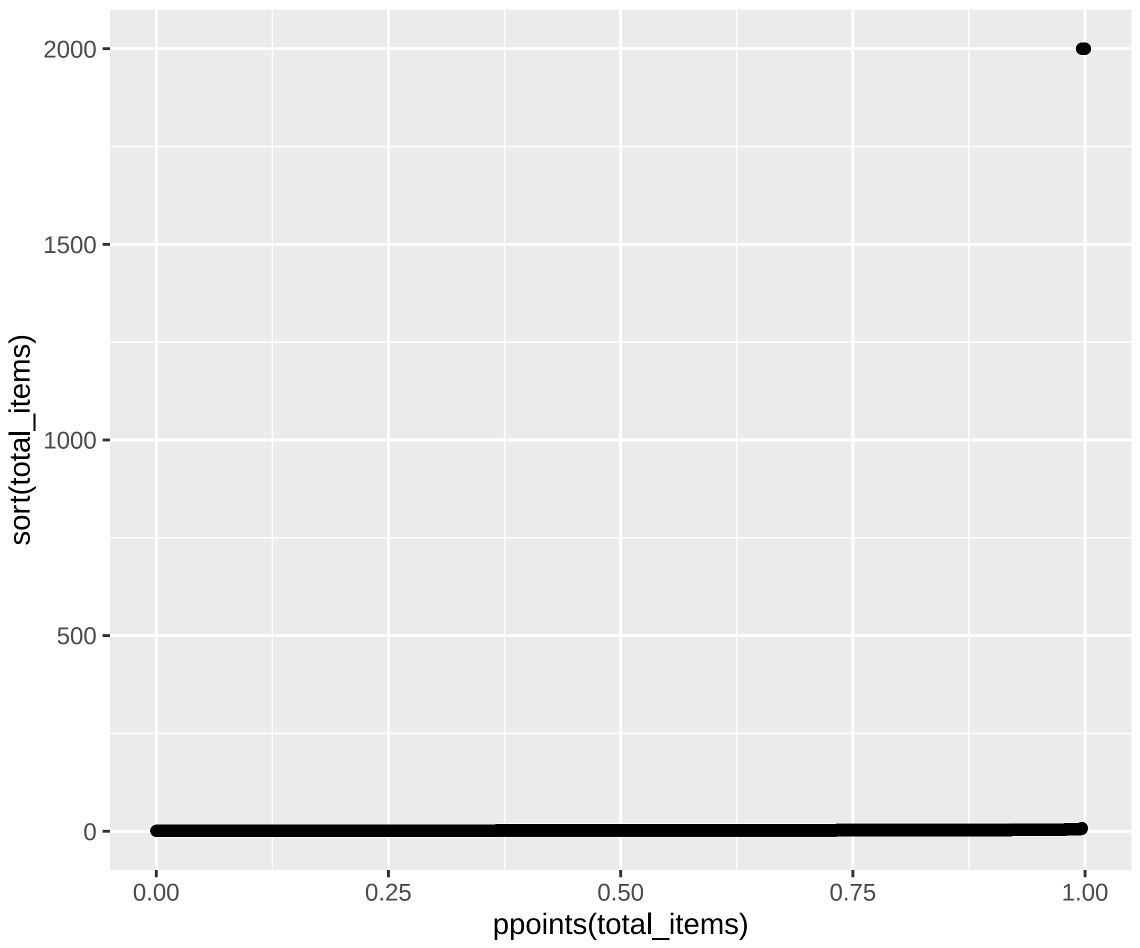


```r
data %>% ggplot(mapping = aes(y=sort(order_amount), x=ppoints(order_amount))) +
         geom_point()
```



From this quick visualization we can clearly see some serious outliers in order amounts, so much so that the majority of the data isn't even really visible and the boxes on the box plot cannot even be seen. Another metric that might be of interest is number of items being ordered in each transaction, again we can visualize this, this time let's just look at the quantile plot:

```r
data %>% ggplot(mapping = aes(y=sort(total_items), x=ppoints(total_items))) +
         geom_point()
```



This plot has uncovered something interesting, there seems to be one large outlier at an order amount of 2000 pairs of sneaker's ordered. Let's take a closer look at this:

```
data %>% filter(total_items == 2000) %>%
         group_by(shop_id, user_id) %>%
         summarise(num_order = n()) %>% as.data.frame()
```

```
##   shop_id user_id num_order
## 1      42     607        17
```

```
data %>% group_by(total_items) %>%
         summarise(num_order = n()) %>% as.data.frame()
```

```
##   total_items num_order
## 1           1      1830
## 2           2      1832
## 3           3       941
## 4           4       293
## 5           5        77
## 6           6         9
## 7           8         1
## 8        2000        17
```

We can see that this large order of 2000 items occurs 17 times and each time it occurs between the same shop and user, this is most likely indicating this transaction is some sort of repeated bulk purchase where the same seller is fulfilling a large order to a repeat buyer who is probably re-selling the sneakers. Outside of the 2000 item transaction we do not see any transactions that would be considered a completely unrealistic order for a buyer for personal (i.e. not reselling) use. We can take a closer look at some of these strange orders:

```
data %>% filter(shop_id == 42 & user_id == 607) %>%
         select(-shop_id, -user_id, -total_items) %>% as.data.frame()
```

```
##    order_id order_amount payment_method          created_at
## 1        16       704000    credit_card 2017-03-07 4:00:00
## 2        61       704000    credit_card 2017-03-04 4:00:00
## 3       521       704000    credit_card 2017-03-02 4:00:00
## 4      1105       704000    credit_card 2017-03-24 4:00:00
## 5      1363       704000    credit_card 2017-03-15 4:00:00
## 6      1437       704000    credit_card 2017-03-11 4:00:00
## 7      1563       704000    credit_card 2017-03-19 4:00:00
## 8      1603       704000    credit_card 2017-03-17 4:00:00
## 9      2154       704000    credit_card 2017-03-12 4:00:00
## 10     2298       704000    credit_card 2017-03-07 4:00:00
## 11     2836       704000    credit_card 2017-03-28 4:00:00
## 12     2970       704000    credit_card 2017-03-28 4:00:00
## 13     3333       704000    credit_card 2017-03-24 4:00:00
## 14     4057       704000    credit_card 2017-03-28 4:00:00
## 15     4647       704000    credit_card 2017-03-02 4:00:00
## 16     4869       704000    credit_card 2017-03-22 4:00:00
## 17     4883       704000    credit_card 2017-03-25 4:00:00
```

I've cut out some variates that we already know, shop_id, user_id and order_amount. One thing that obviously stands out is that these orders are indeed contributing to extremely large order amounts, and as such definitely contribute to the flawed AOV value. Another thing we can see is that the payment method is always credit card and that the time of order is always exactly 4:00:00 for these transactions. These facts further convince me that this is a business venture for this user and they make regular, most likely automated, bulk purchases for reselling purposes.

Another possible source of outliers in order amount would be cases where the price per pair of sneakers sold is very high. Let's look into this:

```
data %>% mutate(pricePerItem = order_amount/total_items) %>%
         group_by(pricePerItem, shop_id) %>%
         summarise(numOrders = n()) %>%
         arrange(desc(pricePerItem)) %>% as.data.frame() %>% head()
```

```
##   pricePerItem shop_id numOrders
## 1        25725      78        46
## 2          352      42        51
## 3          201      12        53
## 4          196      89        61
## 5          195      99        54
## 6          193      50        44
```

It seems that one particular seller has had their price per sneaker recorded as 25,725 for 46 transactions, this is most likely a glitch in the recording of the data. Let's see if this shop has other transactions at a more reasonable price per pair of sneakers:

```
data %>% mutate(pricePerItem = order_amount/total_items) %>%
         group_by(pricePerItem, shop_id) %>%
         summarise(numOrders = n()) %>%
         arrange(desc(pricePerItem)) %>%
         filter(shop_id == 78) %>% as.data.frame()
```

```
##   pricePerItem shop_id numOrders
## 1        25725      78        46
```

It looks like all of this shop's transactions had this un-realistic price per sneaker.

1(a) SUMMARY

To summarize, the given value of the AOV seems unrealistic because when naively calculating it we have ignored two possibilities that could cause outliers in order amount values. The first that we explored was the occurrence of order's being placed that are for a number of sneakers way beyond one would reasonably expect for a regular shopper to purchase and was most likely transactions of a bulk re-seller, not an ordinary customer. The other was odd occurrences of the price paid per item, where we found a certain shop was being recorded as selling pairs of sneakers for a price way beyond what would be reasonable for such an item, this was most likely a glitch in the recording of the data.

A better way to evaluate the data might be to calculate a more robust attribute such that the value would not be skewed by large outliers but still give us an idea of central order values, the median is a potential candidate and is discussed in the next part. Another approach might be to remove the data points that we deem unrealistic or out of the ordinary. An approach in future analyses could be to build a standardized pipeline through which to pass raw data that performs the same exploration as I did above. Such a pipeline would allow us to see odd purchase volumes or price points and allow us to flag such data before we begin our analysis. Recalculating the AOV with these questionable data points omitted is also explored in the next section.

1(b)

Given the requirements I went over at the end of the previous section there may be a couple options we can explore. One metric I would report would be the median value of order amounts, as the median will be a much more robust measure of the central value of the order amounts being less sensitive to the large outliers we found. Another possibility would be a recalculation of the AOV but this time with the questionable data points omitted. From the previous analysis we found that the outliers were being created by two specific instances: the first being shop_id 42 and user_id 607 engaging in very large bulk transactions, and the second being shop_id 78 reporting a very questionable price per pair of sneakers. If we remove the transactions between shop_id 42 and user_id 607 and all transactions by shop_id 78, we may get a much more realistic idea for the mean order amount. I would opt for this rather than a basic trimmed mean approach as a trimmed mean would take a certain percentage of measurements from the top and bottom of the data. I do not wish to do this as we didn't find anything particularly off with the low order amounts or low price points, so I will focus on removing specific odd instances rather than simply naively chopping a percentage of our data away and losing information needlessly.

1(c)

First find the median:

```
medianOrderAmount <- median(data$order_amount)
medianOrderAmount
```

```
## [1] 284
```

The median value is 284. Now let's remove the questionable transactions from the data, and re-calculate the mean:

```
data %>% filter(shop_id != 78) %>%
        filter(shop_id != 42 & user_id != 607) -> cleanedData
```

```
NewAOV <- mean(cleanedData$order_amount)
NewAOV
```

```
## [1] 300.1558
```

The new AOV with the cleaned up data is 300.16. Both of these values seem much more realistic given the nature of the product and are fairly close together meaning we have most likely found a more reasonable price point for the average order amount for sneaker orders.

2(a)

Answer: Speedy Express shipped 54 orders.

Query:

SELECT ShipperName, COUNT(*) AS NumberOfOrders FROM Shippers

JOIN Orders ON Shippers.ShipperID = Orders.ShipperID

WHERE Shippers.ShipperName = "Speedy Express"

GROUP BY Shippers.ShipperID;

2(b)

Answer: The last name of the employee who had the most orders is Peacock, with 40 orders.

Query:

SELECT Employees.LastName, COUNT(*) AS NumberOfOrders FROM Employees

JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID

GROUP BY Employees.LastName

ORDER BY NumberOfOrders DESC

LIMIT 1;

2(c)

For this problem I felt that the request could be interpreted two ways. The first being which item was included in the greatest number of orders for people from Germany (i.e. item ordered the most times), the second was which item had the highest quantity ordered by people in Germany. I thought both may be of interest to analyze in a business context, so I have included queries for both:

Answer, item ordered most times: Gorgonzola Telino, ordered 5 times.

Query:

SELECT Products.ProductName, COUNT(*) AS TimesOrdered FROM Orders

JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID

JOIN Products ON OrderDetails.ProductID = Products.ProductID

JOIN Customers ON Orders.CustomerID = Customers.CustomerID

WHERE Customers.Country = "Germany"

GROUP BY ProductName

ORDER BY TimesOrdered DESC

LIMIT 1;

Answer, item with most quantity ordered: Boston Crab meat, total quantity ordered of 160 units.

Query:

```
SELECT Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantityOrdered FROM Orders
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
JOIN Customers ON Orders.CustomerID = Customers.CustomerID
WHERE Customers.Country = “Germany”
GROUP BY ProductName
ORDER BY TotalQuantityOrdered DESC
LIMIT 1;
```