Avergae order

Camille Bergeron

5/2/2021

Question 1

Clenaing and Viewing

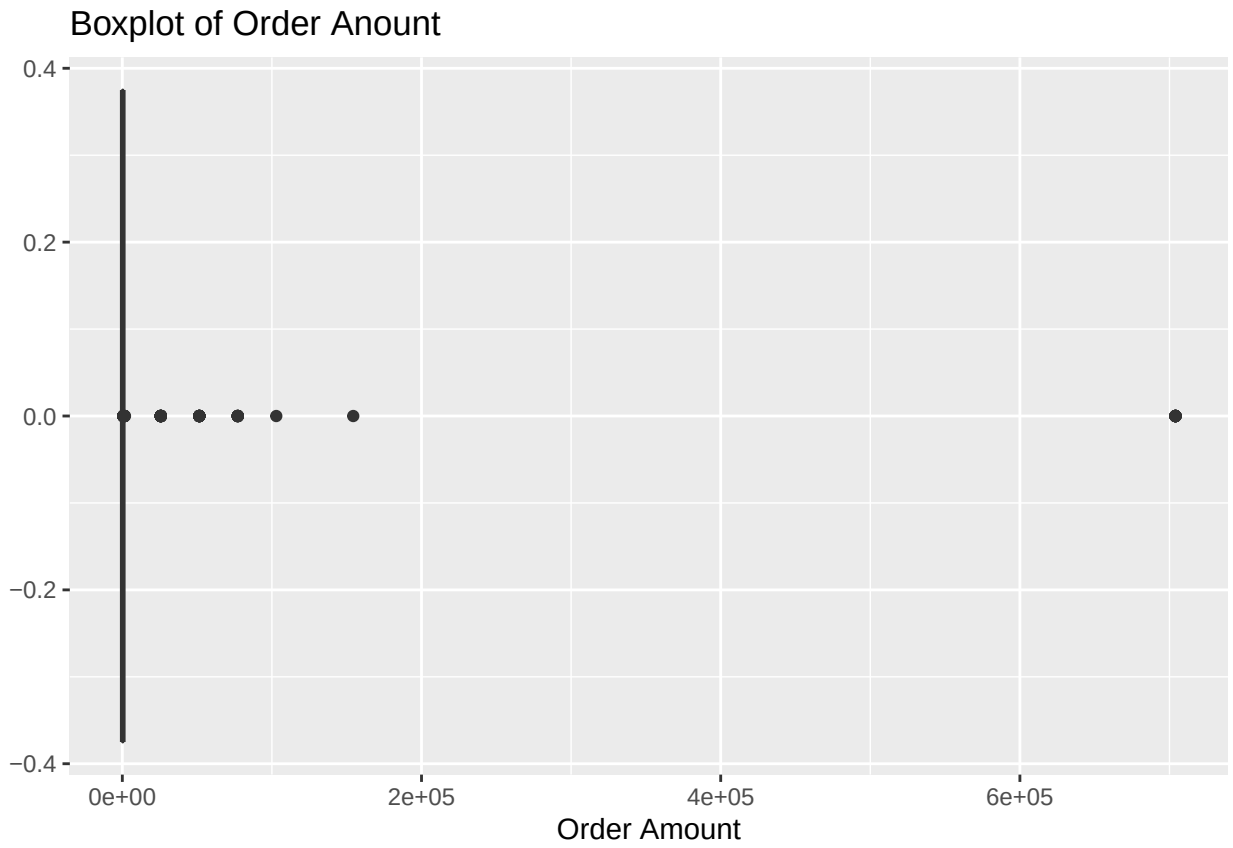
```
# looking at data

# loading the data
orders <- read.csv("2019 Winter Data Science Intern Challenge Data Set - Sheet1.csv")

# avg order value
orders %>%
  select(order_amount) %>%
  summarise(n = mean(order_amount))
```

```
##          n
## 1 3145.128
```

```
# looking at potential outliers
ggplot(orders) +
  geom_boxplot(aes(x = order_amount)) +
  labs(title = "Boxplot of Order Anount", x = "Order Amount")
```




```
# extracting the outliers
boxplot.stats(orders$order_amount)$out
```

```
##   [1] 704000 704000    780    765  25725    780    765    780    780  51450
##  [11]  51450  51450 704000    830  51450    748 154350    772    804    815
##  [21]    885   1056    784  25725 704000    815    885  25725  25725    935
##  [31]  77175 704000   1760   1408  25725  25725 704000  25725   1408    765
##  [41]    736  51450 704000    960 704000    800    804    800    865    745
##  [51]    830    880    920    765    774    790    784 704000  25725 704000
##  [61]    948    845    760    745  51450 102900    965  51450  51450  25725
##  [71]    935  77175    780  77175    805  25725  51450  51450 704000  77175
##  [81]  25725    830 704000   1056    890    980  25725  51450    760  25725
##  [91]  51450    748    786 704000  77175    736    805  25725   1056    736
## [101]    935   1086    736  51450  77175  25725    816    810    740  25725
## [111] 704000  51450   1064  77175    780  51450  51450  77175    735  25725
## [121]    760    880    780    748    748  25725    748    800 704000    780
## [131]  77175    960 704000    790 704000    760  25725    765    880    865
## [141]    772
```

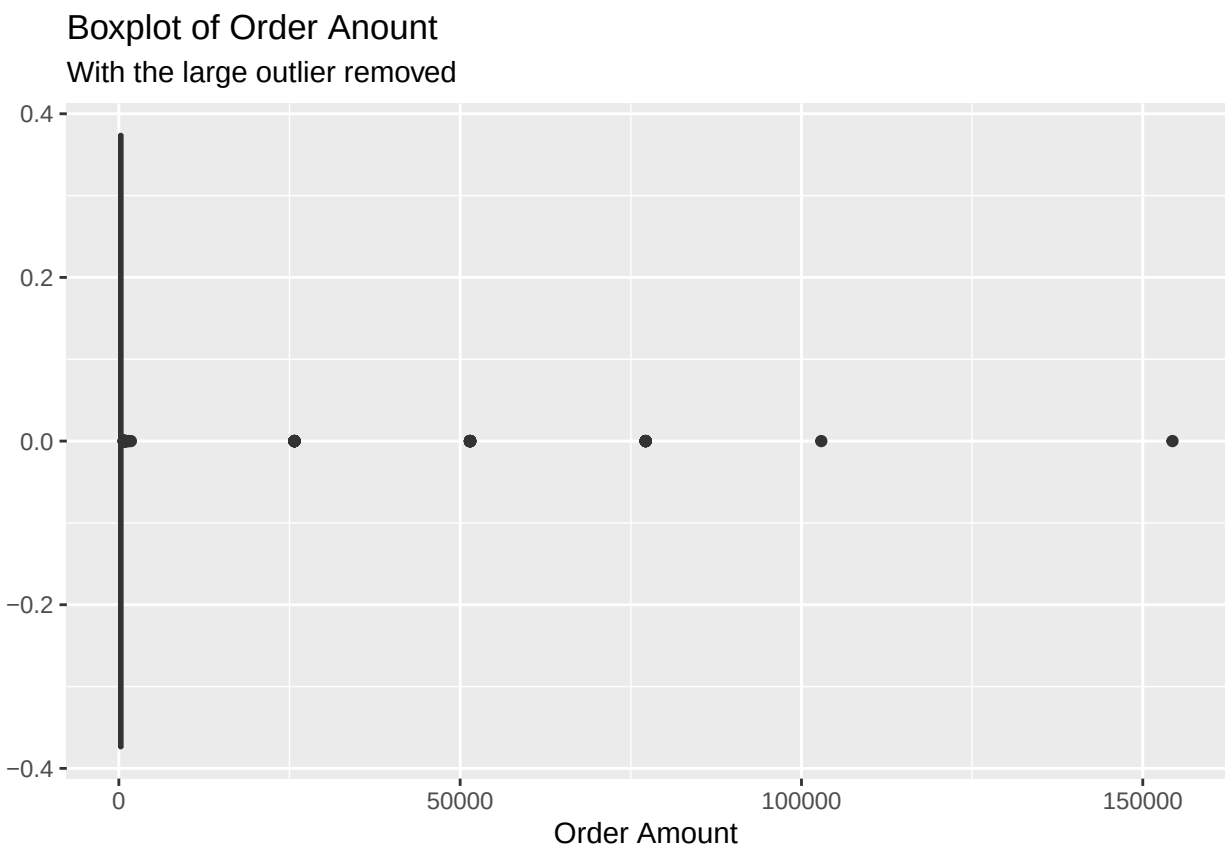
```
orders[which(orders$order_amount == max(orders$order_amount)), ]
```

```
##      order_id shop_id user_id order_amount total_items payment_method
## 16         16      42     607       704000        2000    credit_card
## 61         61      42     607       704000        2000    credit_card
## 521       521      42     607       704000        2000    credit_card
## 1105     1105      42     607       704000        2000    credit_card
## 1363     1363      42     607       704000        2000    credit_card
## 1437     1437      42     607       704000        2000    credit_card
```

```
## 1563     1563      42     607       704000        2000    credit_card
## 1603     1603      42     607       704000        2000    credit_card
## 2154     2154      42     607       704000        2000    credit_card
## 2298     2298      42     607       704000        2000    credit_card
## 2836     2836      42     607       704000        2000    credit_card
## 2970     2970      42     607       704000        2000    credit_card
## 3333     3333      42     607       704000        2000    credit_card
## 4057     4057      42     607       704000        2000    credit_card
## 4647     4647      42     607       704000        2000    credit_card
## 4869     4869      42     607       704000        2000    credit_card
## 4883     4883      42     607       704000        2000    credit_card
##              created_at
## 16   2017-03-07 4:00:00
## 61   2017-03-04 4:00:00
## 521  2017-03-02 4:00:00
## 1105 2017-03-24 4:00:00
## 1363 2017-03-15 4:00:00
## 1437 2017-03-11 4:00:00
## 1563 2017-03-19 4:00:00
## 1603 2017-03-17 4:00:00
## 2154 2017-03-12 4:00:00
## 2298 2017-03-07 4:00:00
## 2836 2017-03-28 4:00:00
## 2970 2017-03-28 4:00:00
## 3333 2017-03-24 4:00:00
## 4057 2017-03-28 4:00:00
## 4647 2017-03-02 4:00:00
## 4869 2017-03-22 4:00:00
## 4883 2017-03-25 4:00:00
```

The largest outliers are all exactly $704000 by the same user at the same shop with the same payment method. The only difference is that the transactions take place on different days in March, but all at 4:00 am.

```r
# new boxplot without the large order
orders %>%
  filter(user_id != 607) %>%
  ggplot() +
  geom_boxplot(aes(x = order_amount)) +
  labs(title = "Boxplot of Order Anount", x = "Order Amount", subtitle = "With the large outlier remove
```




```
  # there are still lot of outliers so these clearly not it

# looking at this average
orders %>%
  filter(user_id != 607) %>%
  summarise(n = mean(order_amount))
```

```
##          n
## 1 754.0919
```

```
# this is much cheaper, so maybe this is better?

# separating the date and time
orders <- orders %>%
  separate(created_at, c("date", "time"), " ") %>%
  mutate(date = as.Date(date))

# taking out the dates to see if they were closing out
orders %>%
  filter(shop_id == 42) %>%
  arrange(date)
```

```
##    order_id shop_id user_id order_amount total_items payment_method       date
## 1      2019      42     739          352           1          debit 2017-03-01
## 2      2492      42     868          704           2          debit 2017-03-01
## 3      4422      42     736          704           2    credit_card 2017-03-01
## 4       521      42     607       704000        2000    credit_card 2017-03-02
## 5      4647      42     607       704000        2000    credit_card 2017-03-02
```

```
## 6      2988      42     819         1056          3          cash 2017-03-03
## 7        61      42     607       704000       2000   credit_card 2017-03-04
## 8       410      42     904          704          2   credit_card 2017-03-04
## 9      4232      42     962          352          1          cash 2017-03-04
## 10     2767      42     970          704          2   credit_card 2017-03-05
## 11       16      42     607       704000       2000   credit_card 2017-03-07
## 12     1912      42     739          704          2          cash 2017-03-07
## 13     2298      42     607       704000       2000   credit_card 2017-03-07
## 14      836      42     819          704          2          cash 2017-03-09
## 15     3999      42     886          352          1         debit 2017-03-09
## 16     1365      42     797         1760          5          cash 2017-03-10
## 17      309      42     770          352          1   credit_card 2017-03-11
## 18     1437      42     607       704000       2000   credit_card 2017-03-11
## 19     4626      42     809          352          1   credit_card 2017-03-11
## 20      980      42     744          352          1         debit 2017-03-12
## 21     1472      42     907         1408          4         debit 2017-03-12
## 22     2154      42     607       704000       2000   credit_card 2017-03-12
## 23     3698      42     839          352          1         debit 2017-03-12
## 24     3904      42     975          352          1         debit 2017-03-12
## 25      939      42     808         1056          3   credit_card 2017-03-13
## 26     1368      42     926         1408          4          cash 2017-03-13
## 27     4768      42     720          704          2   credit_card 2017-03-14
## 28     1363      42     607       704000       2000   credit_card 2017-03-15
## 29     4327      42     788          704          2         debit 2017-03-16
## 30     1603      42     607       704000       2000   credit_card 2017-03-17
## 31     1930      42     770          352          1   credit_card 2017-03-17
## 32     1563      42     607       704000       2000   credit_card 2017-03-19
## 33     2054      42     951          352          1         debit 2017-03-19
## 34     1521      42     756          704          2         debit 2017-03-22
## 35     4869      42     607       704000       2000   credit_card 2017-03-22
## 36     2610      42     868          704          2         debit 2017-03-23
## 37       41      42     793          352          1   credit_card 2017-03-24
## 38     1105      42     607       704000       2000   credit_card 2017-03-24
## 39     1513      42     946          352          1         debit 2017-03-24
## 40     3333      42     607       704000       2000   credit_card 2017-03-24
## 41     3514      42     726         1056          3         debit 2017-03-24
## 42     3652      42     830          352          1   credit_card 2017-03-24
## 43     4295      42     859          704          2          cash 2017-03-24
## 44     4746      42     872          352          1         debit 2017-03-24
## 45      835      42     792          352          1          cash 2017-03-25
## 46     4883      42     607       704000       2000   credit_card 2017-03-25
## 47     2004      42     934          704          2          cash 2017-03-26
## 48     2274      42     747          704          2         debit 2017-03-27
## 49     2836      42     607       704000       2000   credit_card 2017-03-28
## 50     2970      42     607       704000       2000   credit_card 2017-03-28
## 51     4057      42     607       704000       2000   credit_card 2017-03-28
##         time
## 1  12:42:26
## 2  18:33:33
## 3  12:19:49
## 4   4:00:00
## 5   4:00:00
## 6   9:09:25
## 7   4:00:00
```

```
## 8  14:32:58
## 9   0:01:19
## 10 10:45:42
## 11  4:00:00
## 12  5:42:52
## 13  4:00:00
## 14 14:15:15
## 15 20:10:41
## 16  6:28:21
## 17 18:14:39
## 18  4:00:00
## 19  8:21:26
## 20 13:09:04
## 21 23:00:22
## 22  4:00:00
## 23  2:45:09
## 24  1:28:31
## 25 23:43:45
## 26  2:38:34
## 27 10:26:08
## 28  4:00:00
## 29 23:37:57
## 30  4:00:00
## 31  8:11:13
## 32  4:00:00
## 33 11:49:12
## 34 13:10:31
## 35  4:00:00
## 36 18:10:14
## 37 14:15:41
## 38  4:00:00
## 39 13:35:04
## 40  4:00:00
## 41 17:51:05
## 42 22:26:58
## 43 20:50:40
## 44  0:57:24
## 45 21:31:25
## 46  4:00:00
## 47  9:21:26
## 48 20:48:19
## 49  4:00:00
## 50  4:00:00
## 51  4:00:00
```

```
# this is the average amount spent on items
orders %>%
  mutate(avg = order_amount / total_items) %>%
  summarise(n = mean(avg))
```

```
##          n
## 1 387.7428
```

Another way to do analysis similar to the average order value (AOV) is by looking at the overage value per item. This number is $387 per sneaker.

Question 2

Part a: Orders Shipped by Speedy Express

Notes: - Speedy Express is ShipperID 1

SQL Query

SELECT COUNT(*) FROM [Orders] WHERE ShipperID = (SELECT ShipperID FROM [Shippers] WHERE Shippers.ShipperName = “Speedy Express”)

There are 54 orders shipped by Speedy Express

Part b: Last Name of Employee with Most Orders

SQL Query

SELECT CustomerName FROM [Customers] WHERE CustomerID = (SELECT CustomerID FROM [Orders] GROUP BY CustomerID ORDER BY COUNT(CustomerID) DESC LIMIT 1)

The last name of the customer with the most orders shipped is “Handel”. The actual output of the previous code is “Ernst Handel”.

Part c: Product Orderded moost from customers in Germany

SQL Query

- this gets the proper customers from Germany

SELECT CustomerID FROM [Customers] WHERE Country = “Germany”

- this gets the associated orders from those customers

SELECT OrderID FROM [Orders] WHERE CustomerID IN (SELECT CustomerID FROM [Customers] WHERE Country = “Germany”)

- this gets the associated Product ids from the order identified order ids

SELECT ProductID FROM [OrderDetails] WHERE OrderID IN (SELECT OrderID FROM [Orders] WHERE CustomerID IN (SELECT CustomerID FROM [Customers] WHERE Country = “Germany”))

- this gets the most purchased productID

SELECT ProductID FROM [OrderDetails] WHERE OrderID IN (SELECT OrderID FROM [Orders] WHERE CustomerID IN (SELECT CustomerID FROM [Customers] WHERE Country = “Germany”)) GROUP BY ProductID ORDER BY COUNT(ProductID) DESC LIMIT 1

- this grabs the product name, and is the final query

SELECT ProductName FROM [Products] WHERE ProductID = (SELECT ProductID FROM [OrderDetails] WHERE OrderID IN (SELECT OrderID FROM [Orders] WHERE CustomerID IN (SELECT CustomerID FROM [Customers] WHERE Country = “Germany”)) GROUP BY ProductID ORDER BY COUNT(ProductID) DESC LIMIT 1)

The most purchases product by customers in Germany is called Gorgonzola Telino.