Shopify

```
#import the dataset
library(readr)
library(Hmisc)
library(ggplot2)
library(dplyr)
shopify <- read_csv("/Users/ray/Downloads/shopify.csv")
```

Question 1

```
summary(shopify$order_amount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      90     163     284    3145     390  704000
```

```
sd(shopify$order_amount)
```

```
## [1] 41282.54
```

We see that the mean is 3145, while the median is 284 and standard deviation is 41282.54, which is quite large comparing to the mean. While the third quartile is at 390, the maximum is at 704000. The gap between the two numbers is huge. Those could be the reason why the calculation could gone wrong.

```
#check if there's any missing value in the dataset
which(is.na(shopify))
```

```
## integer(0)
```

```
#check for values larger than 3145
sample(shopify[which(shopify$order_amount > 3145),],5)
```

```
## # A tibble: 63 x 5
##    order_id created_at          total_items shop_id order_amount
##       <dbl> <chr>                     <dbl>   <dbl>        <dbl>
##  1       16 2017-03-07 4:00:00         2000      42       704000
##  2       61 2017-03-04 4:00:00         2000      42       704000
##  3      161 2017-03-12 5:56:57            1      78        25725
##  4      491 2017-03-26 17:08:19           2      78        51450
##  5      494 2017-03-16 21:39:35           2      78        51450
##  6      512 2017-03-09 7:23:14            2      78        51450
##  7      521 2017-03-02 4:00:00         2000      42       704000
```

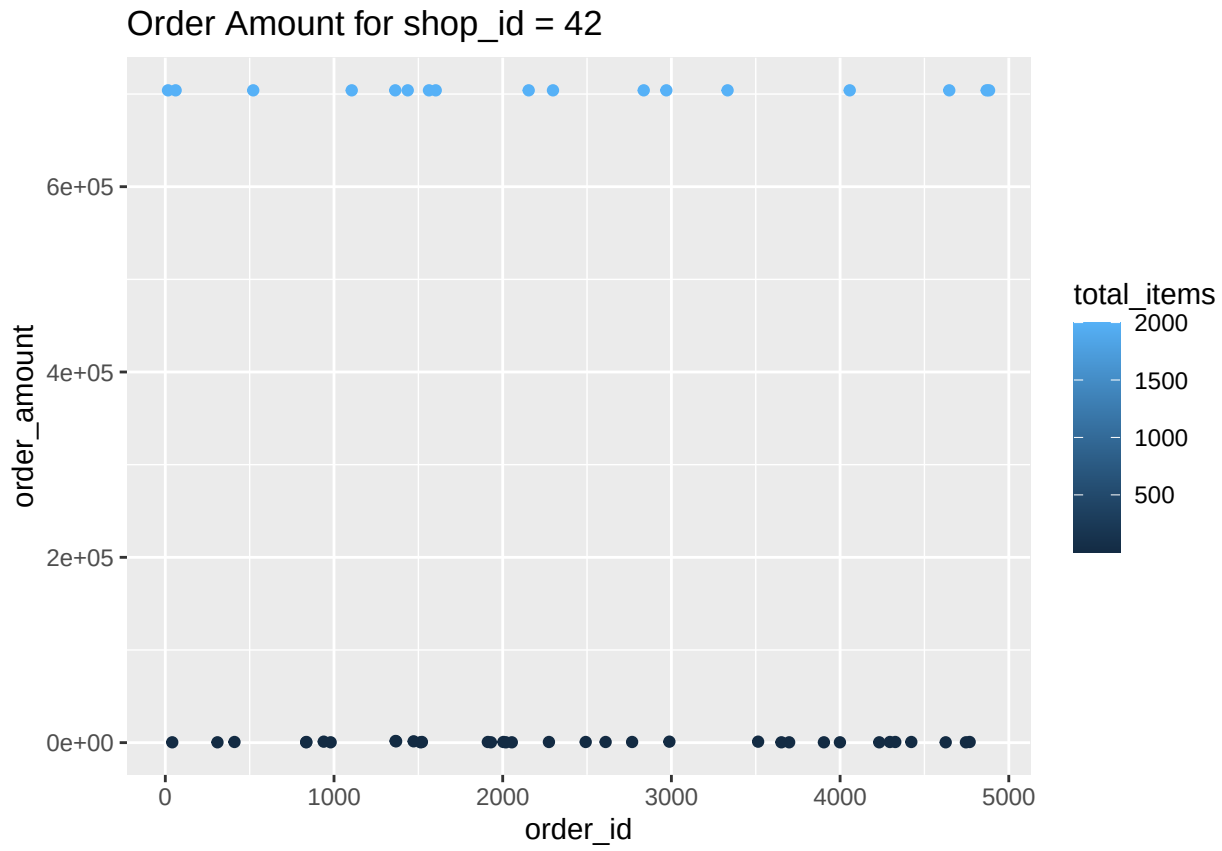
```
##  8      618 2017-03-18 11:18:42           2      78        51450
##  9      692 2017-03-27 22:51:43           6      78       154350
## 10     1057 2017-03-15 10:16:45           1      78        25725
## # ... with 53 more rows
```

```
unique(shopify[which(shopify$order_amount > 3145),]$shop_id)
```

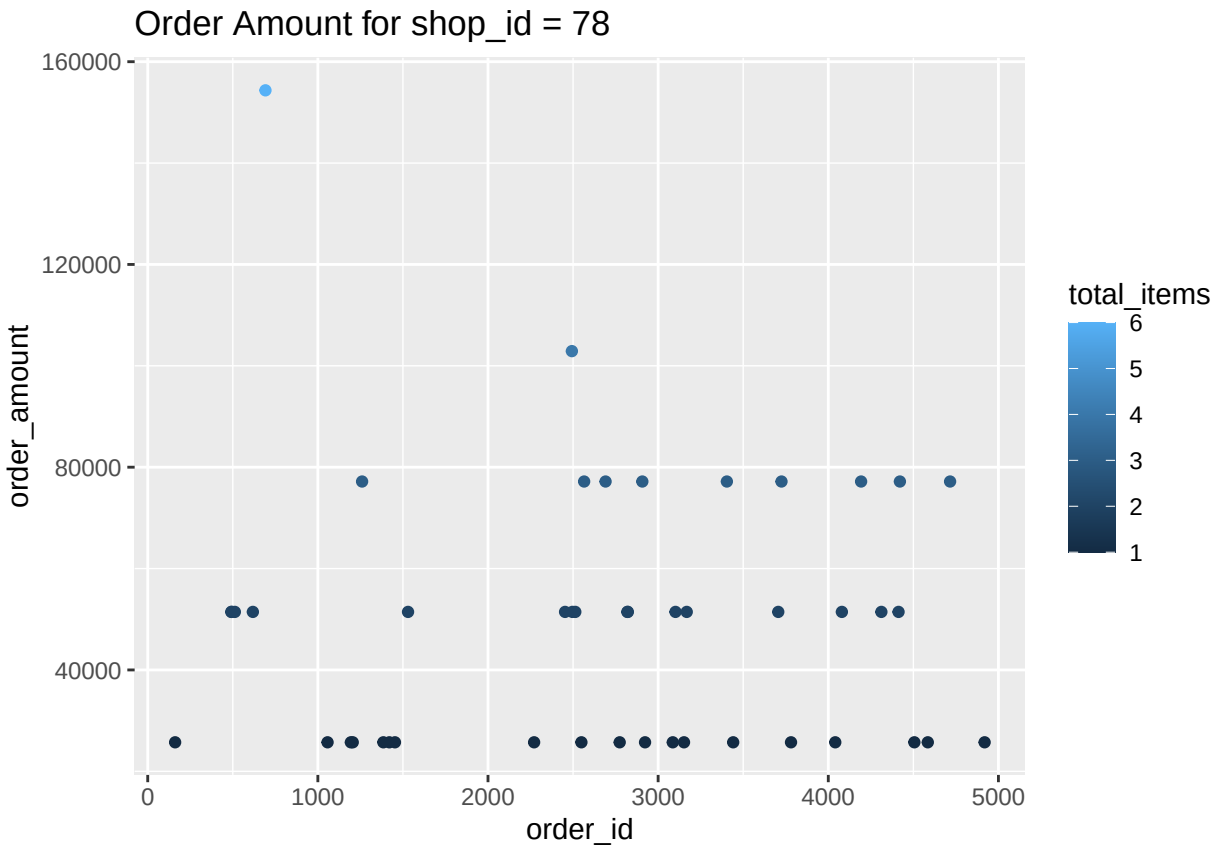
```
## [1] 42 78
```

We can see that shop_id 42 and 78 are the two main stores that may cause outlier in our dataset.

```
shop42 = as.data.frame(shopify[which(shopify$shop_id == 42),])
ggplot(shop42, aes(y = order_amount, x = order_id, color=total_items))+ geom_point() + ggtitle("Order Am
```




```
shop78 = as.data.frame(shopify[which(shopify$shop_id == 78),])
ggplot(shop78, aes(y = order_amount, x = order_id, color=total_items))+ geom_point() + ggtitle("Order Am
```



We could see from the table and graph that shop42 usually sells a bulk order of items (e.g. 2000 items with total of 704000), while shop78 usually sells a few items (e.g. 1 item with total of 25725).

```
shop42 %>%
  group_by(total_items) %>%
  summarise(total=sum(order_amount))
```

```
## # A tibble: 6 x 2
##   total_items    total
##         <dbl>    <dbl>
## 1           1     5280
## 2           2     9152
## 3           3     3168
## 4           4     2816
## 5           5     1760
## 6        2000 11968000
```

```
shop78 %>%
  group_by(total_items) %>%
  summarise(total=sum(order_amount))
```

```
## # A tibble: 5 x 2
##   total_items  total
##         <dbl>  <dbl>
## 1           1 488775
```

```
## 2           2 823200
## 3           3 694575
## 4           4 102900
## 5           6 154350
```

Therefore, these relatively "few" or "massive" observations generated by store ID: 42 and 78 skew the distribution order_amount. Some better ideas on evaluating the data would be divide the groups up into different regions based on the order_amount. For example, larger than 3000 and smaller than 500, etc. Or we could evaluate based on the total_items sold.

We need to get a better understanding of what the shopify managers want to know from the dataset and what are the important factors we should highlight from the analysis. However, it is always imporrant to note that we cannot be looking at a single metric in isolation, as it can be extremely misleading.

```
shopify %>%
  group_by(total_items) %>%
  summarise(total=sum(order_amount),
            mean=mean(order_amount),
            median = median(order_amount),
            `standard deviation` = sd(order_amount))
```

```
## # A tibble: 8 x 5
##   total_items    total    mean median `standard deviation`
##         <dbl>    <dbl>   <dbl>  <dbl>                <dbl>
## 1           1   763777    417.    153                2593.
## 2           2  1374394    750.    306                4761.
## 3           3  1120803   1191.    459                7471.
## 4           4   277672    948.    592                5978.
## 5           5    58470    759.    765                 161.
## 6           6   161460  17940     948               51154.
## 7           8     1064   1064    1064                  NA
## 8        2000 11968000 704000  704000                   0
```

Dividing up the dataset based on total_items could give us a clearer idea of what we might want. Also, the median could provide additional information about the order_amount, allowing store managers to better understand the evaluation. Some important thing to note is that there's only 1 order with total_item of 8, so the sd is NA. All order_amouont for total_items of 2000 is the same, so the sd is 0.

Question 2

part a

part b

part c

SQL Statement:

```
SELECT *, COUNT(Orders.ShipperID) AS total
FROM Orders
Inner Join Shippers ON Shippers.ShipperID == Orders.ShipperID
group by Orders.ShipperID
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 3

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID	ShipperName	Phone	total
10249	81	6	1996-07-05	1	Speedy Express	(503) 555-9831	54
10250	34	4	1996-07-08	2	United Package	(503) 555-3199	74
10248	90	5	1996-07-04	3	Federal Shipping	(503) 555-9931	68

Figure 1: There are total of 54 orders shipped by Speedy Express.

```
SELECT LastName, count(Orders.EmployeeID) AS total
FROM [Orders]
INNER JOIN Employees ON Employees.EmployeeID = Orders.EmployeeID
Group By Orders.EmployeeID
Order By total DESC
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 9

LastName	total
Peacock	40
Leverling	31
Davolio	29
Callahan	27
Fuller	20
Suyama	18
King	14
Buchanan	11
Dodsworth	6

Figure 2: The last name of the employee with most orders is Peacock.

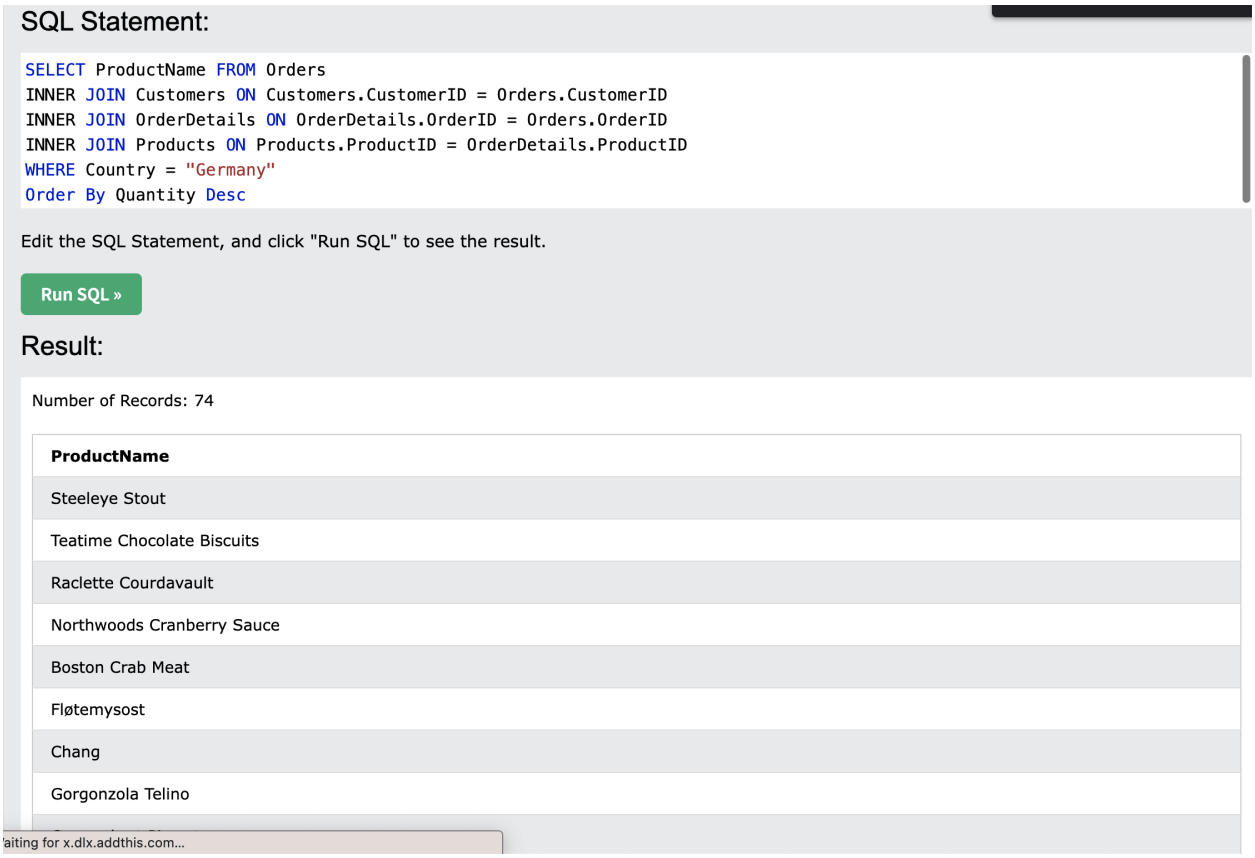


Figure 3: The product named Steeleye Stout was ordered most by customers in Germany.