# Shopify Challenge

David Ding

21/09/2021

## Question 1

**Part a)** The mean, as a measure of central tendency, is incredibly sensitive to outliers in the data. Assuming that the calculation for average order value is correct, we can investigate the presence of outliers within the dataset.

```
df <- read_csv("shopify_data.csv")

summary(df$order_amount)
```

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

```
df %>% filter(order_amount > 10000)
```

```
## # A tibble: 63 x 7
##    order_id shop_id user_id order_amount total_items payment_method created_at
##       <dbl>   <dbl>   <dbl>        <dbl>       <dbl> <chr>          <chr>     
##  1       16      42     607       704000        2000 credit_card    2017-03-07 ~
##  2       61      42     607       704000        2000 credit_card    2017-03-04 ~
##  3      161      78     990        25725           1 credit_card    2017-03-12 ~
##  4      491      78     936        51450           2 debit          2017-03-26 ~
##  5      494      78     983        51450           2 cash           2017-03-16 ~
##  6      512      78     967        51450           2 cash           2017-03-09 ~
##  7      521      42     607       704000        2000 credit_card    2017-03-02 ~
##  8      618      78     760        51450           2 cash           2017-03-18 ~
##  9      692      78     878       154350           6 debit          2017-03-27 ~
## 10     1057      78     800        25725           1 debit          2017-03-15 ~
## # ... with 53 more rows
```

From this output, we immediately see that there are orders for $704000. Although plausible, Lebron James and his teammates' shoe orders don't reflect the population at large!

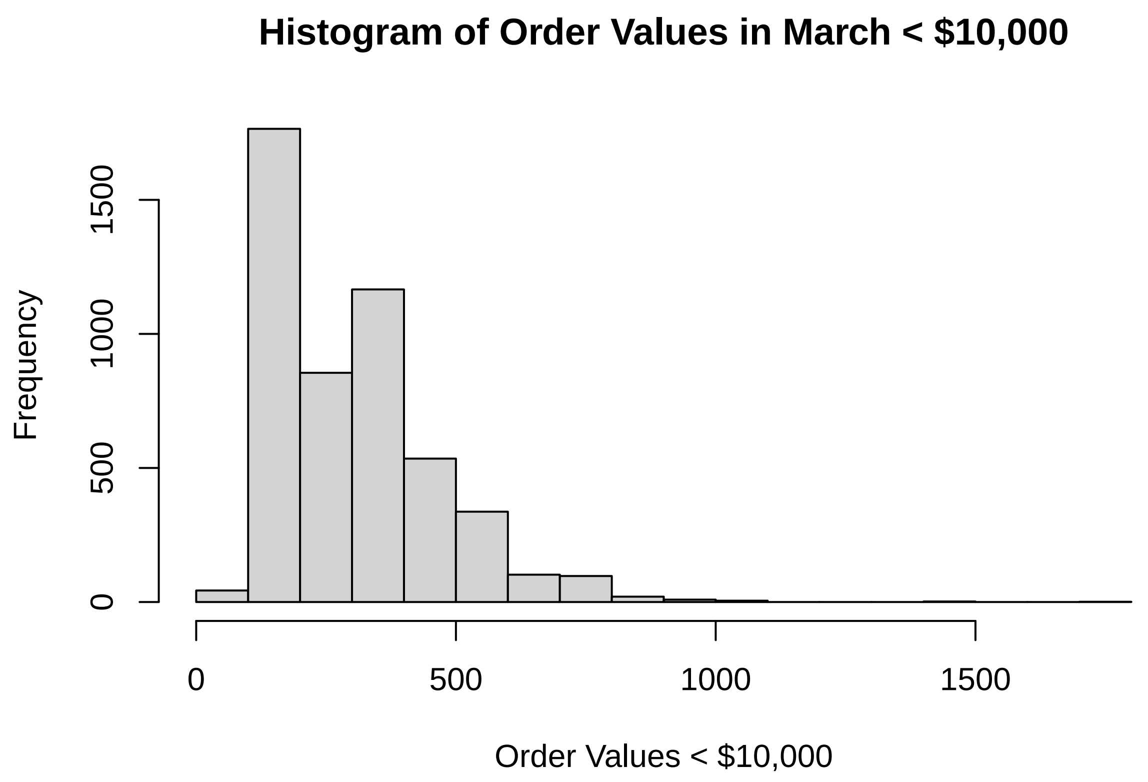
We see that there are 63 orders with an order value of over $10,000. And there are 44 orders with an order value of over $50,000!

This dataset needs to be cleaned of outliers before any measures of central tendency are reported. We cannot remove outliers the "normal" way (2 Standard Deviations away from the mean), so we might rely on the median of this dataset as a good measure of centrality.

**Part b)** Although we could report on the median, being a more robust measure of central tendency. We need to verify that the data is unimodal. We can quickly verify that by plotting all the points less than $10000:

```
df_filtered <- df %>% filter(order_amount < 10000)
```

```r
hist(df_filtered$order_amount,
     xlab = "Order Values < $10,000",
     main = "Histogram of Order Values in March < $10,000")
```



This amount is **relatively** unimodal. However, the natural question begins to form: "What is the average shoe price sold during the month of March"

To combine both these ideas, we can ask the question:

"What is the median shoe price during the month of March" and report the findings there:

```r
df_cleaned <- df %>% mutate(avg_price = order_amount / total_items)


median(df_cleaned$avg_price)
```

**Part c)**

```
## [1] 153
```

Thus, we can report that the median price of a shoe being sold on Shopify in March is $153.00.

**Question 2:**

**Part a)**

```
SELECT
    count(*),
    s.ShipperName
FROM
    Orders o
LEFT JOIN
  Shippers s
ON o.ShipperID = s.ShipperID
WHERE s.ShipperName = "Speedy Express";
```

We obtain 54 orders shipped by Speedy Express.

As an alternative solution, we can "store" the ID of "Speedy Express" as a variable, then inner join back onto the Orders table. This should improve efficiency when we have a larger table.

```
WITH s AS(
SELECT
    ShipperID
FROM
    Shippers
WHERE ShipperName = "Speedy Express"
)
SELECT
    count(*)
FROM
    Orders
INNER JOIN s ON Orders.ShipperID = s.ShipperID;
```

**Part b)** We have a more "roundabout" way of doing it here. This is again due to efficiency. We want to compute the top employee in sales, then store it temporarily. We do this instead of joining all orders on all employees, which is potentially computationally expensive.

```
WITH top_employee_id AS(
SELECT
    count(*) as n_orders,
    EmployeeID
FROM
    ORDERS
GROUP BY EmployeeID
ORDER BY n_orders desc
LIMIT 1
)
SELECT
    LastName
FROM
    employees
INNER JOIN top_employee_id
ON employees.employeeID = top_employee_id.employeeID;
```

The top employee Last Name is "Peacock".

**Part c)**

```
WITH german_customers AS (
  SELECT
      CustomerID
  FROM [Customers]
  WHERE Country = "Germany"
),
german_orders AS(
  SELECT
      OrderID
  FROM
      ORDERS
  INNER JOIN german_customers
      ON ORDERS.CustomerID = german_customers.CustomerID
),
top_product as(
  SELECT
      productid,
      sum(quantity) as n_products
  FROM
      OrderDetails
  INNER JOIN german_orders
    ON OrderDetails.OrderId = german_orders.OrderId
  GROUP BY productID
  ORDER BY n_products desc
  LIMIT 1
)
SELECT
    ProductName
FROM
    Products
INNER JOIN top_product
ON top_product.productid = Products.productID;
```

The most popular product from German orders is Boston Crab Meat