Shopify W21 Challenge By Geerthika Senthilmanoharan

Question 1: Given some sample data, write a program to answer the following: click here to access the required data set

On Shopify, we have exactly 100 sneaker shops, and each of these shops sells only one model of shoe. We want to do some analysis of the average order value (AOV). When we look at orders data over a 30 day window, we naively calculate an AOV of $3145.13. Given that we know these shops are selling sneakers, a relatively affordable item, something seems wrong with our analysis.

a. Think about what could be going wrong with our calculation. Think about a better way to evaluate this data.

```
#load library and data
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tibble)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
shopData <- read.csv('data/Data_Science_Intern_Challenge_Data_Set.csv')
head(shopData)
```

```
##   order_id shop_id user_id order_amount total_items payment_method
## 1        1      53     746          224           2           cash
## 2        2      92     925           90           1           cash
## 3        3      44     861          144           1           cash
```

```
## 4        4      18     935          156           1    credit_card
## 5        5      18     883          156           1    credit_card
## 6        6      58     882          138           1    credit_card
##            created_at
## 1 2017-03-13 12:36:56
## 2 2017-03-03 17:38:52
## 3  2017-03-14 4:23:56
## 4 2017-03-26 12:43:37
## 5  2017-03-01 4:35:11
## 6 2017-03-14 15:25:01
```

```
#Getting a quick summary of the data
summary(shopData)
```

```
##     order_id       shop_id          user_id       order_amount
##  Min.   :   1   Min.   :  1.00   Min.   :607.0   Min.   :    90
##  1st Qu.:1251   1st Qu.: 24.00   1st Qu.:775.0   1st Qu.:   163
##  Median :2500   Median : 50.00   Median :849.0   Median :   284
##  Mean   :2500   Mean   : 50.08   Mean   :849.1   Mean   :  3145
##  3rd Qu.:3750   3rd Qu.: 75.00   3rd Qu.:925.0   3rd Qu.:   390
##  Max.   :5000   Max.   :100.00   Max.   :999.0   Max.   :704000
##   total_items       payment_method      created_at
##  Min.   :   1.000   Length:5000        Length:5000
##  1st Qu.:   1.000   Class :character   Class :character
##  Median :   2.000   Mode  :character   Mode  :character
##  Mean   :   8.787
##  3rd Qu.:   3.000
##  Max.   :2000.000
```

```
#Frequency of order sizes
freqTab <- as.data.frame(table(shopData$total_items))
names(freqTab)[1] = 'Total_Items_Bought'
freqTab
```

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

There appear to be many transactions of size 2000; this is most likely the contributor to why the original average order value was so large.

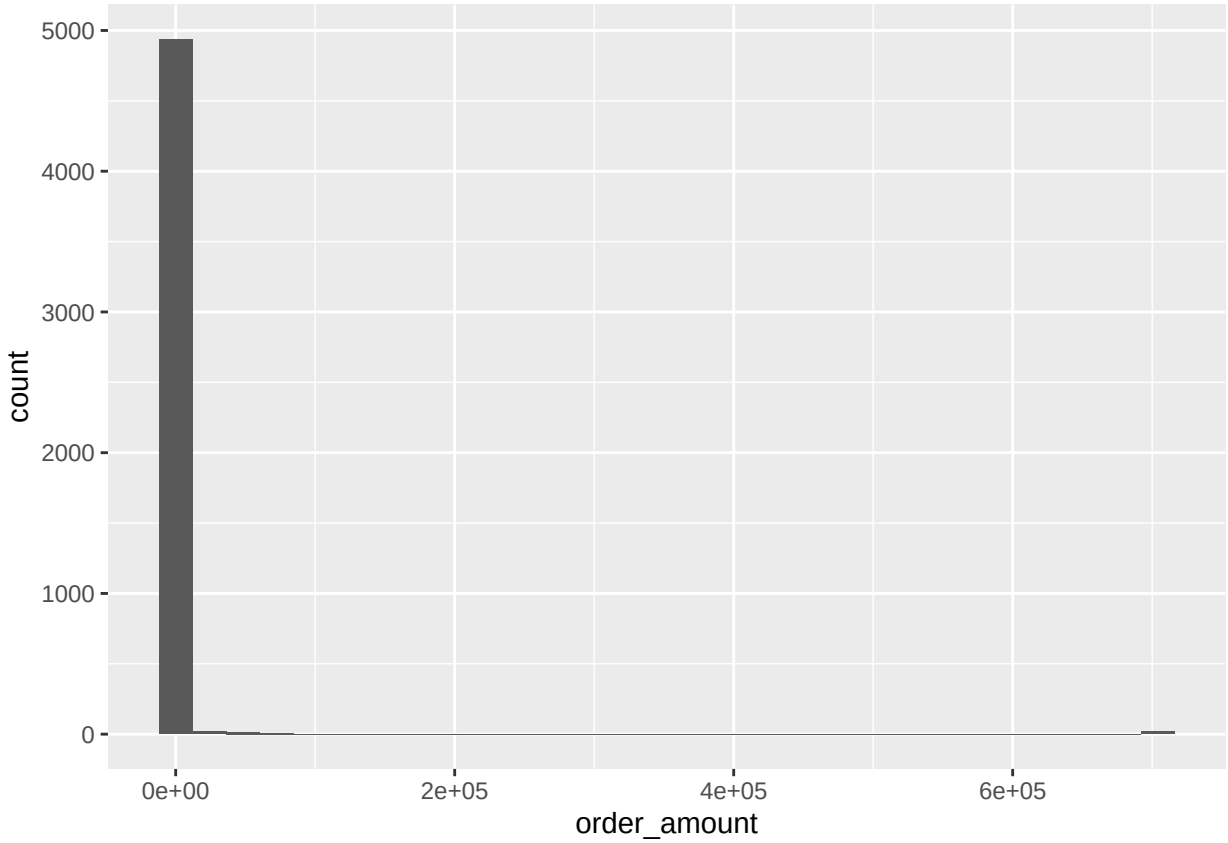
```
# further looking at the large transactions
bigOrder <- shopData %>%
  filter(shopData$total_items==2000)
bigOrder$created_at <- ymd_hms(bigOrder$created_at)
bigOrder <- bigOrder[order(bigOrder$created_at),]
bigOrder
```

```
##    order_id shop_id user_id order_amount total_items payment_method
## 3       521      42     607       704000        2000    credit_card
## 15     4647      42     607       704000        2000    credit_card
## 2        61      42     607       704000        2000    credit_card
## 1        16      42     607       704000        2000    credit_card
## 10     2298      42     607       704000        2000    credit_card
## 6      1437      42     607       704000        2000    credit_card
## 9      2154      42     607       704000        2000    credit_card
## 5      1363      42     607       704000        2000    credit_card
## 8      1603      42     607       704000        2000    credit_card
## 7      1563      42     607       704000        2000    credit_card
## 16     4869      42     607       704000        2000    credit_card
## 4      1105      42     607       704000        2000    credit_card
## 13     3333      42     607       704000        2000    credit_card
## 17     4883      42     607       704000        2000    credit_card
## 11     2836      42     607       704000        2000    credit_card
## 12     2970      42     607       704000        2000    credit_card
## 14     4057      42     607       704000        2000    credit_card
##             created_at
## 3  2017-03-02 04:00:00
## 15 2017-03-02 04:00:00
## 2  2017-03-04 04:00:00
## 1  2017-03-07 04:00:00
## 10 2017-03-07 04:00:00
## 6  2017-03-11 04:00:00
## 9  2017-03-12 04:00:00
## 5  2017-03-15 04:00:00
## 8  2017-03-17 04:00:00
## 7  2017-03-19 04:00:00
## 16 2017-03-22 04:00:00
## 4  2017-03-24 04:00:00
## 13 2017-03-24 04:00:00
## 17 2017-03-25 04:00:00
## 11 2017-03-28 04:00:00
## 12 2017-03-28 04:00:00
## 14 2017-03-28 04:00:00
```

Upon further analysis, we see that these purchases were all made by user_id 607 at shop_id at 4:00 pm on different days. This tells me that a bug/glitch is causing these entries to show up in the data, or an automated purchase is being made.

```
# Histogram of order amount frequency
ggplot(data = shopData) +
  geom_histogram(aes(x = order_amount))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



When graphing the frequency of the order amount, the histogram appeared to be skewed to the right.

b. What metric would you report for this dataset?

From the above analysis, I concluded that the best metric to analyze this data is the median, as it will give a better representation of the middle value.

c. What is its value?

```
median(shopData$order_amount)
```

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

Question 2: For this question you'll need to use SQL. Follow this link to access the data set required for the challenge. Please use queries to answer the following questions. Paste your queries along with your final numerical answers below.

a. How many orders were shipped by Speedy Express in total?

SELECT COUNT(*) FROM Orders WHERE ShipperID = 1;

Answer: 54

Thought Process: From the Shippers table, I determined that the ShipperID for Speedy Express was 1, so I used a where clause to filter the data. Since each row in the orders table represented a new order, I used COUNT to tally the number of rows.

b. What is the last name of the employee with the most orders?

SELECT e.LastName FROM Employees e JOIN (SELECT MAX (numOrders) , EmployeeID FROM (SELECT EmployeeID, COUNT(*) numOrders FROM [Orders] GROUP BY EmployeeID)) t ON e.EmployeeID = t.EmployeeID;

Answer: Peacock

Thought Process: First, I used GROUP BY to group the rows in the Orders table by the employee id. I then extracted the max value and the employee id associated with it using a select statement to create a table. I then joined the table that I had just created with the Employees table on the employee id to extract the employee's last name with the maximum number of orders.

c. What product was ordered the most by customers in Germany?

SELECT ProductName FROM Products a JOIN (SELECT MAX (numProds) , ProductID FROM (SELECT COUNT(*) numProds, d.ProductID FROM Orders o JOIN OrderDetails d ON o.OrderID = d.OrderID
JOIN CUSTOMERS c ON o.CustomerID = c.CustomerID AND c.Country = 'Germany' GROUP BY d.ProductID)) b ON a.ProductID = b.ProductID;

Answer: Gorgonzola Telino

Thought Process: First, I created a sub-query that calculated the total number of each ordered product using a GROUP BY. I joined in the Order Details table to get the product id information and joined in the customer table to filter the country to show only orders from Germany. I then extracted the max value and the product id associated with it using a select statement to create a table. I then joined the table that I had just created with the Products table on the product id to extract the product name of the product most ordered by customers in Germany.