Winter 2022 Challenge

Michael He

9/20/2021

Task 1

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.

Answers 1A

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

A quick glimpse of the dataset shows some massive bulk orders, which can significantly distort the average order value. In particular, Shop 42 sells bulk orders worth 704,000 dollars to user 607. This is a big outlier we should filter from our AOV calculation. The new adjusted AOV is about 754 dollars. This is much less than 3145, but is still atypical for online sneaker orders.

We also want to look at the selling prices, in case some sneakers are much more expensive than other brands. For example, Shop 78 sells very expensive sneakers worth 25,725 dollars each pair and the second most expensive shoe only costs 352 dollars! So orders from Shop 78 are also outliers in a way. If we filter out Shop 78 in addition to Shop 42, then the new AOV is 302.6 dollars. This is now more typical of a sneaker ecommerce platform that's not solely focused on rare items or bulk orders.

```
library(googlesheets4)

# download the spreadsheet completely
gs4_deauth()
sneakershop <- read_sheet("16i38oonuX1y1g7C_UAmiK9GkY7cS-64DfiDMNiR41LM",
sheet="Sheet1",)
```

```
## v Reading from "2019 Winter Data Science Intern Challenge Data Set".
```

```
## v Range ''Sheet1''.
```

```
# a quick ordering of order in descending dollar amount
sneakershop %>%
  arrange(desc(order_amount))
```

```
## # A tibble: 5,000 x 7
##    order_id shop_id user_id order_amount total_items payment_method
##       <dbl>   <dbl>   <dbl>        <dbl>       <dbl> <chr>
##  1       16      42     607       704000        2000 credit_card
##  2       61      42     607       704000        2000 credit_card
##  3      521      42     607       704000        2000 credit_card
##  4     1105      42     607       704000        2000 credit_card
```

```
##  5     1363      42     607       704000        2000 credit_card
##  6     1437      42     607       704000        2000 credit_card
##  7     1563      42     607       704000        2000 credit_card
##  8     1603      42     607       704000        2000 credit_card
##  9     2154      42     607       704000        2000 credit_card
## 10     2298      42     607       704000        2000 credit_card
## # ... with 4,990 more rows, and 1 more variable: created_at <dttm>
```

```
# filtering out the 700k orders, then find AOV
sneakershop %>%
  filter(order_amount < 160000) %>%
  summarize(AOV = mean(order_amount))
```

```
## # A tibble: 1 x 1
##     AOV
##   <dbl>
## 1  754.
```

```
# find out which shoes are expensive with price_level
sneakershop %>%
  mutate(price_level = order_amount / total_items) %>%
  filter(order_amount < 160000) %>%
  filter(price_level < 25725) %>%
  summarize(AOV = mean(order_amount))
```

```
## # A tibble: 1 x 1
##     AOV
##   <dbl>
## 1  303.
```

Part 1B & 1C

What metric would you report for this dataset? What is its value?

I would look at the median order amount, because it shows how much customers in the middle of the shopping spectrum spent. Here the median order amount is 284 dollars and the median total items bought is 2. This makes sense, as most online shoppers usually buy only a few pairs of sneakers at a time and sneaker prices generally fall within the low 3-digit range.

```
# find the median of order_amount and total_items ordered
sneakershop %>%
  summarize(MedOrd = median(order_amount), MedItem = median(total_items))
```

```
## # A tibble: 1 x 2
##   MedOrd MedItem
##    <dbl>   <dbl>
## 1    284       2
```

Task 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.

https://www.w3schools.com/SQL/TRYSQL.ASP?FILENAME=TRYSQL_SELECT_ALL

Answer Part 2A

How many orders were shipped by Speedy Express in total?

We first go to Orders, because it contains the shipping data. We double check with Shippers to make sure Speedy Express's Shipper ID is 1. THe SQL query and result are below. There are 54 orders shipped by Speedy Express.

```
# the SQL query is below

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

# result:
# COUNT(ShipperID)
# 54
```

Answer Part 2B

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

We first go to Orders, because it contains the employee data. We run the SQL query with a COUNT function and the result says Employee 4 ships the most orders (since the query is in descending order). We check with Employees and find Margaret Peacock as Employee number 4. So Peacock is the last name of the employee with the most orders.

```
# SELECT EmployeeID
# FROM Orders
# GROUP BY EmployeeID
# ORDER BY COUNT(*) DESC
# LIMIT    1;

# result:
# EmployeeID
# 4
```

Answer Part 2C

What product was ordered the most by customers in Germany?

We need to work with multiple tables. We first join the column CustomerID from Customers to the Orders table. Then we join the OrderID column from Order to the OrderDetails table. We join the ProductID column from the OrdersDetails table to the Products table. Now we have all the columns and variables together.

We use a WHERE statement to filter out customers from Germany, group all the names of the products German customers have ordered, and calculate the total of each product (at the top of code chunk). Now we can order the total quantity per item in a descending order. Boston Crab Meat is the number one selling item in Germany, with 160 units!

```
# SELECT ProductName,
# SUM(Quantity) AS SumAmt

# FROM Customers
# LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
```

```
# LEFT JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
# LEFT JOIN Products ON OrderDetails.ProductID = Products.ProductID

# WHERE Customers.Country = 'Germany'
# GROUP BY Products.ProductName
# ORDER BY SumAmt DESC
# LIMIT 1

# result:
# ProductName   SumAmt
# Boston Crab Meat  160
```