Winter 2021 Data Science Intern Challenge

Christiana Koebel

crkoebel@uwaterloo.ca

Question 1: Given some sample data, write a program to answer the following:

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.

By doing some preliminary analysis on the dataset, I determined that the Average Order Value of $3145.13 was calculated by computing the mean of the order_amount column across all shops:

```
import pandas as pd
from pandas import ExcelWriter
from pandas import ExcelFile

shoes = pd.read_excel('/home/crkoebel/intern_challenge/2019 Winter Data
   Science Intern Challenge Data Set.xlsx')

# Average Order Value (AOV)

average_order_value = (shoes['order_amount']).mean() # AOV calculation
print(average_order_value)
>> 3145.128
```

A problem with this is that the calculation does not account for differences between shops that might contribute to certain shops selling more products than others and at difference prices. For example, some stores might offer incentives to customers such as free shipping, coupons, or free products for spending a minimum amount of money. Simply finding the average order amount does not take these differences into account. Usually, average order value is calculated separately by store; as such, this metric on its own is not appropriate for this analysis.

A better way to evaluate this data is to calculate the average amount of money spent on a pair of sneakers per order across shops. This is better because it incorporates both the average amount of money a customer spends per order, and the average number of items sold per order.

b. What metric would you report for this dataset?

For this dataset, I will report a metric that uses Average Order Value (AOV) as well as Average Basket Value (ABV). AOV is the average amount of money each customer spends per transaction (as calculated in part a.), and ABV is the average number of items sold per transaction. To get the average amount of money spent per pair of shoes (Average Shoes Value ASV), I will divide the AOV by the ABV:

ASV = AOV/ABV

However, before I calculate this metric, I want to investigate the dataset further. To look at the amount of money spent per shoes item on each order, I created a new column in the dataset:

```
shoes['amt_per_item'] = shoes['order_amount']/shoes['total_items']
# Create column: order amount per item for each order

mx = max(shoes['amt_per_item'])
mn = min(shoes['amt_per_item'])

print(mx)
>> 25725
print(mn)
>> 90

print(shoes[shoes['amt_per_item'] == mx][['order_id', 'shop_id',
                                          'amt_per_item']].head())
# shop_id = 78. Outlier, data entry error, or luxury shoe model?
>>
order_id  shop_id  amt_per_item
161       78       25725.0
491       78       25725.0
494       78       25725.0
512       78       25725.0
618       78       25725.0

print(len(shoes[shoes['amt_per_item'] == mx]))
>> 46
```

Looking at the range of this new column, we see that the maximum amount spent per item in an order is $25,725, while the minimum is $90. This is a huge difference. In fact, the second highest is $352, which is still significantly less than the maximum of $25,725. Filtering the dataset to see which shop's shoes sell for $25,725 reveals that shop_id 78 had 46 orders with an average of $25,725 per pair of shoes. This stuck out to me and made me wonder if this was an outlier, data entry error, or if this shop simply sells a luxury model of shoe. Since shop 78's shoes are significantly more money than the other shops', I decided to calculate two Average Shoes Values (ASV); one including shop 78 and one excluding shop 78, just in case there was some sort of issue with the pricing of its shoes.

c. What is its value?

Using the unfiltered dataset, the AOV is $3145.13, the ABV is 8.79, and the Average Shoes Value (ASV) is $357.92:

```
# Average Order Value (AOV)
average_order_value = (shoes['order_amount']).mean() # AOV calculation
print(average_order_value)
>> 3145.128

# Average Basket Value (ABV)
average_basket_value = (shoes['total_items']).mean() # ABV calculation
print(average_basket_value)
>> 8.7872

# Average Shoes Value (ASV)
average_shoes_value = average_order_value/average_basket_value
print(average_shoes_value)
>> 357.92152221412965
```

This means that the average amount of money a customer spends on a pair of sneakers in an order is $357.92, which is a more useful metric than AOV in this context.

If we remove shop 78 from the dataframe (since its sneakers are $25,725 per pair on average, which is significantly more than other shops), the AOV is $2717.37, the ABV is 8.85, and the Average Shoes Value (ASV) is $307.01.

```
# Remove shop_id 78 from dataframe

shoes2 = shoes[shoes['shop_id'] != 78]

# New Average Order Value (AOV)
average_order_value_2 = (shoes2['order_amount']).mean()
print(average_order_value_2)
>> 2717.3677836092047

# New Average Basket Value (ABV)
average_basket_value_2 = (shoes2['total_items']).mean()
print(average_basket_value_2)
>> 8.851029471134437

# New Average Shoes Value (ASV)
average_shoes_value_2 = average_order_value_2/average_basket_value_2
print(average_shoes_value_2)
>> 307.01149425287355
```

Thus, customers spend an average of $307.01 per pair of sneakers if we exclude shop 78.

Question 2: For this question you'll need to use SQL. 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?

To solve this, we need the Orders and Shippers tables. Since the Orders table does not have the shippers' names, it needs to be joined by ShipperID with the Shippers table, which has both shippers' names and IDs. With this new table, we can count the number of orders shipped by Speedy Express by using the COUNT() function on OrderID, the GROUP BY clause on ShipperName, and the HAVING clause for ShipperName = 'Speedy Express'.

```
SELECT COUNT(OrderID) AS NumOrders
FROM Orders
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID
GROUP BY ShipperName
HAVING ShipperName = 'Speedy Express'

>> 54
```

Therefore, the number of orders shipped by Speedy Express is: 54.

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

To answer this question, I first determined which tables are needed: Employees (for the LastName and EmployeeID fields) and Orders (for the EmployeeID and OrderID fields).

To break my solution into more manageable pieces, I created a table of the number of orders for each employee by using the COUNT() function on OrderID and the GROUP BY clause on EmployeeID. This will make the final query less complex, and might be useful in solving other questions for the company.

```
CREATE TABLE EmployeeNumOrders AS
SELECT EmployeeID, COUNT(OrderID) AS NumOrders
FROM Orders
GROUP BY EmployeeID
ORDER BY NumOrders DESC
```

Next, using this new table, we can find the highest number of orders out of all employees using the MAX() function, and the last name of the corresponding employee.

```
SELECT LastName
FROM Employees
WHERE EmployeeID =
(SELECT EmployeeID
FROM EmployeeNumOrders
WHERE NumOrders IN (
SELECT MAX(NumOrders)
FROM EmployeeNumOrders))

>> Peacock
```

Running this query indicates that the last name of the employee with the most orders is: Peacock.

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

To approach this question, I began by identifying the tables I need: Customers (for the CustomerID and Country fields), Orders (for the CustomerID and OrderID fields), OrderDetails (for the OrderID, ProductID and Quantity fields), and Products (for the ProductID and ProductName fields).

First, I created a table of orders only from customers in Germany by using the Orders and Customers tables:

```
CREATE TABLE GermanyOrders AS
SELECT OrderID
FROM Orders
WHERE CustomerID IN(
SELECT CustomerID
FROM Customers
WHERE Country = 'Germany')
```

By joining this new table with OrderDetails and utilizing the SUM() function on Quantity and the GROUP BY clause on ProductID, I created a table showing the total quantity of products ordered for each ProductID by customers in Germany.

```
CREATE TABLE ProductQuantities AS
SELECT ProductID, SUM(Quantity) AS TotalQuantity
FROM OrderDetails
LEFT JOIN GermanyOrders ON OrderDetails.OrderID = GermanyOrders.OrderID
GROUP BY ProductID
ORDER BY TotalQuantity DESC
```

Lastly, by using ProductName and ProductID from the Product table, ProductID from the new ProductQuantities table, along with the MAX() function on total product quantity, we can find the name of the product ordered the most by customers in Germany.

```
SELECT ProductName
FROM Products
WHERE ProductID =
(SELECT ProductID
FROM ProductQuantities
WHERE TotalQuantity IN (
SELECT MAX(TotalQuantity)
FROM ProductQuantities))

>> Gorgonzola Telino
```

The result of this query reveals that the product ordered the most by customers in Germany was: Gorgonzola Telino.