Winter 2021 Data Science Intern Challenge

Please complete the following questions, and provide your thought process/work. You can attach your work in a text file, link, etc. on the application page. Please ensure answers are easily visible for reviewers!

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.

Thought Process:

- I have no experience with google apps script (I'm guessing by writing code you mean in google apps script. I may be wrong). However, I will try my hardest to explain it in words.
- Data set goes from March 1, 2017, to March 30, 2017
- Three types of payments: Cash (1594), debit (1671), and credit (1735). Totaling (5000)
- Total Items sold (averages between 1-5, not counting outlier of 2000)
- Interestingly, each UserID of 607 has the shopID of 42 (which is the outlier with 2000 items sold).
- Also, each ShopID of 78 sells 2-4 sneakers for $51 450- $102 900 (2x51450). Since not only 1 customer, also, many customers buy the product.

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

UserID of 607 is an outlier in this situation. The person buys 2000 sets of sneakers costing around $704, 000. Which averages to about $352/sneaker, this is adequate. Additionally, ShopID 78 sells 2-4 sneakers for $51 450- $102 900. This brings up the question that if there is only 1 model of shoe available in each 100 sneaker shops, then why is ShipID 78 selling for a

greater price. Could it be the customer is buying something else rather than a sneaker, or it could be a glitch in the system?

I believe the data should be separated. The separation should be between the extremely high payments (>$1000) and the other regular payments. This allows the average order value (AOV) of regular payments (which average about 1-5 sneakers brought) to be in the margins. For the special case, it would be beneficial to investigate further for the sneaker type, who paid, etc.

b. What metric would you report for this dataset?

I initially did not know what metrics meant, however after some research, I figured it out. According to DZone, “Metrics capture a value pertaining to your systems at a specific point in time; for example, the number of users currently logged into the database.” It is a sort of monitoring tool that is required to have the database run smoother. Therefore, the metric I would report for this database is the ShopID of 78 and the UserID of 607. Specifically, when these ID’s buy/sell sneakers. It would also be good to understand what different variations of the sneaker are brought/sold. This data would help in understanding what time intervals are the sneakers being brought/sold.

c. What is its value?

The value we would be looking for is anything above the selling price of $1000/sneaker. Specifically, anything in the 10000’s or 100000’s.

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.

Note: I looked through all the tables and marked the PRIMARY KEY and where it is a foreign key so that it would be easier when I write the query.

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

Query	Result
SELECT COUNT (*) AS "Total shipped by Speedy Express" FROM Orders JOIN Shippers	54

ON Orders.ShipperID = Shippers.ShipperID WHERE Orders.ShipperID = 1;	

Thought Process:

1. When reading the question, two words stood out (orders, Speedy Express). These were keyword which pointed me towards the two tables (Orders, Shippers). I checked the two tables, saw that the PRIMARY KEY ShipperID is a foreign key in Orders. Then I saw that Speedy Express has a ShipperID of 1. Then I knew that I will need to count the total orders where ShipperID was 1. Hence, I joined the two tables with ShipperID where it was 1.

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

Query	Result
SELECT Employees.EmployeeID AS "Employee ID", Employees.LastName AS "Employee Last Name", COUNT(Orders.CustomerID) AS "Number of Orders" FROM Employees JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID GROUP BY 1 ORDER BY 3 DESC;	Last name: Peacock # of orders: 40

Thought Process:

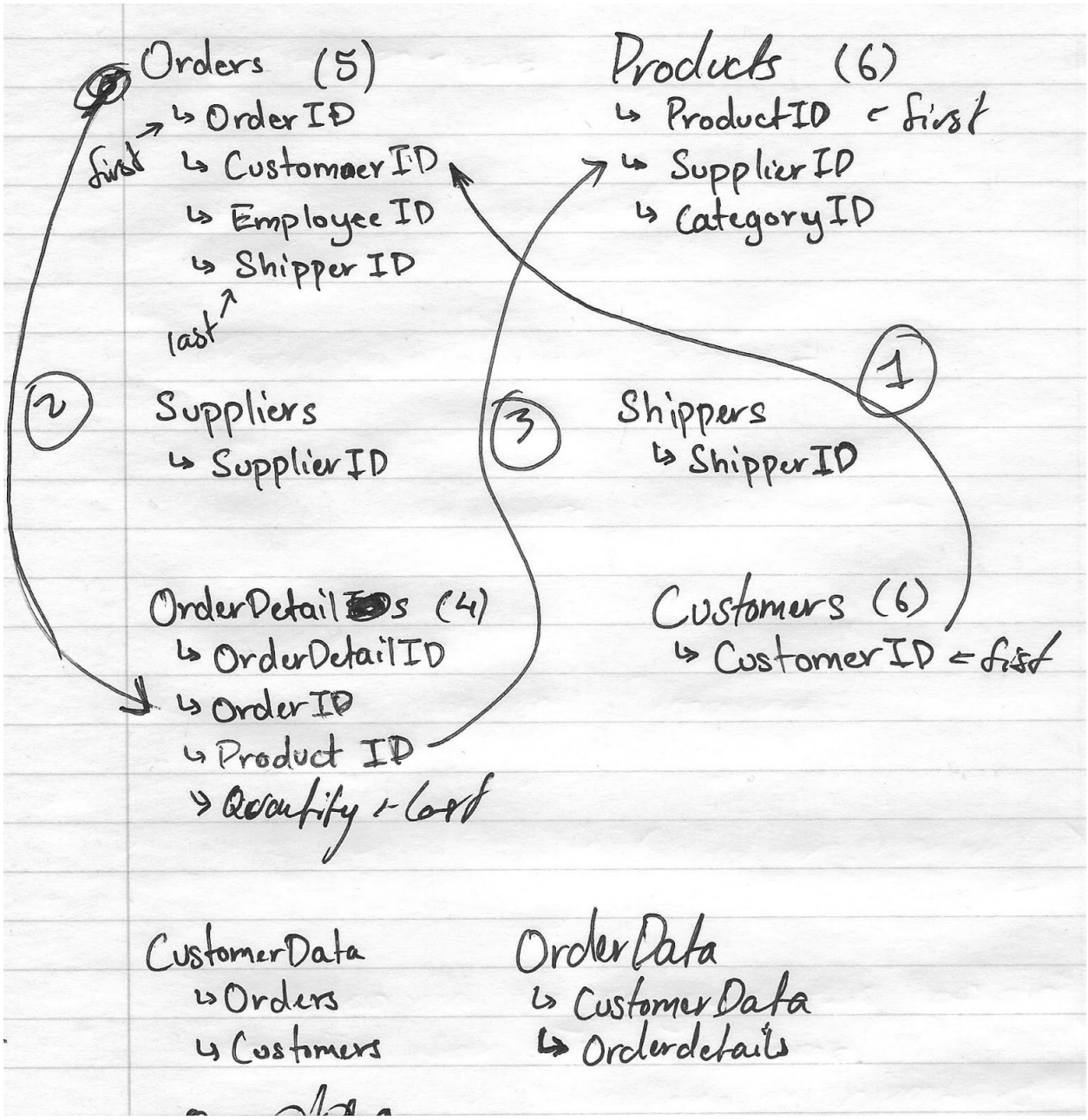
1. Same as above, read question, saw key words (last name, employee, most orders). Checked the tables (employees, orders), saw EmployeeID as PRIMARY KEY. Needed to count orders, group by the Employees and order them from highest to lowest.

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

Query	Result
WITH CustomerData AS (SELECT Orders.OrderID As "Order ID", Orders.CustomerID As "Customer ID", Customers.CustomerName AS "Customer Name", Customers.Country As "Country" FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID WHERE Customers.Country = "Germany"), OrderData AS (	Product: Lakkalikööri # Ordered: 4

SELECT OrderDetails."ProductID",
  COUNT (CustomerData."Customer ID") AS "Number Of Products",
  CustomerData.Country
FROM OrderDetails
JOIN CustomerData
  ON OrderDetails.OrderID = CustomerData."Order ID"
GROUP BY 1
ORDER BY 1 DESC
)
 	
SELECT Products.ProductName AS "Product Name",
  OrderData."Number Of Products" AS "Products Shipped"
FROM Products
JOIN OrderData
  ON Products.ProductID = OrderData.ProductID;	

Thought Process:


Orders (5)
↳ OrderID
↳ CustomaerID
↳ EmployeeID
↳ ShipperID
first
last
Products (6)
↳ ProductID ← first
↳ SupplierID
↳ CategoryID
Suppliers
↳ SupplierID
Shippers
↳ ShipperID
1
2
3
OrderDetailIDs (4)
↳ OrderDetailID
↳ OrderID
↳ Product ID
↳ Quantity ← last
Customers (6)
↳ CustomerID ← first
CustomerData
↳ Orders
↳ Customers
OrderData
↳ CustomerData
↳ Orderdetails