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.

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

> Shop_ID = 42 and Shop_ID = 78 have anomalously high total order values of $11,990,176 and $2,263,800. These enormous values skew the average order value (AOV) to the $3145.13 mark quoted by your analyst.
>
> The high order values in Shop_ID = 42 are explained by the fact that User_ID = 607 ordered 2000 pairs of sneakers at $352 each, multiple times this month. The amount, price per order and number of orders are high enough to indicate a wholesale operation. If user 607 can make a profit on so many expensive sneakers it may be worth investigating what market has such demand for sneakers and how we can best exploit it.
>
> The high order values in Shop_ID = 78 are even more suspicious as multiple users appear to be buying sneakers priced at $25,725 per pair. With User_ID = 834 spending $102,900 in a single purchase of four items. This also requires further review.
>
> If the transactions at Shop_ID 42 and 78 are fraudulent or incorrect they still warrant exploration.
>
> By removing the outliers, Shop_ID 42 and 78, we find the AOV to be a reasonable $300.16. We can report that number and hold off on reporting on shops 42 and 78 pending further investigation.

Work and thought process:

> I uploaded the table to PgAdmin and used PSQL to analyze the table because it makes analysis quick and convenient. The steps I took are below.
>
> 1. Confirming one-month time frame
>
> SELECT MAX(created_at), MIN(created_at)

```
FROM sales;
```

2. Confirming average order value

```
SELECT ROUND(AVG(order_amount),2)
FROM sales;
```

3. Number of sales and total order value by store

Shows stop_id 42 and 78 having anomalously high total order value

```
SELECT
        shop_id,
        SUM(total_items) as "number_of_items" ,
        SUM(order_amount) as total_order_value,
FROM sales
GROUP BY 1
ORDER BY 3 DESC;
```

4. Average order value by store

```
SELECT shop_id, ROUND(AVG(order_amount),2) average_order_value
FROM sales
GROUP BY 1
ORDER BY 2 DESC;
```

5. Finding what the average order value is for all stores excluding 42 and 78 which are outliers

```
SELECT ROUND(AVG(order_amount),2) as average_order_value
FROM (SELECT shop_id, order_amount
        FROM sales
        WHERE shop_id != 42 and shop_id!= 78
        ORDER BY 2 DESC) AS sales;
```

6. Exploring the purchases made in stores 42 and 78

```
SELECT
        shop_id,
        order_id,
        user_id,
        order_amount,
        total_items,
        order_amount / total_items AS price_per_item
FROM sales
WHERE shop_id = 42 OR shop_id = 78
ORDER BY price_per_item DESC;
```

b. What metric would you report for this dataset?

If the outliers were not removed the median would be the more appropriate metric for characterizing the central value of the skewed data set.

c. What is its value?

$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?

54 orders were shipped by Speedy Express.

```
SELECT COUNT (*)
FROM Orders
LEFT JOIN Shippers
        ON Orders.ShipperID = Shippers.ShipperId
WHERE ShipperName = 'Speedy Express';
```

I joined the left tables that contained the information I was looking for to make sure that the most populous table set the standard for the following joins and no data was lost. I limited the query to rows containing 'Speedy Express' as the shipper. Then I counted the rows that each had one order to show the number of orders.

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

Peacock is the employee with the most orders

```
SELECT LastName
FROM (
        SELECT Employees.EmployeeID, Employees.LastName, COUNT(OrderID)
        FROM Orders
        LEFT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
        GROUP BY 1
        ORDER BY 3 DESC
        LIMIT 1);
```

Like the previous example I "left joined" all the tables I would need to answer the question to keep the orders table stable. Then I grouped the table by EmployeeID to reduce the problem of duplicate names, ordered the query in descending order and limited the query to the row with the highest value. From that subquery I selected the last name to answer the question.

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

The most frequently ordered product by customers in Germany is Gorgonzola Telino

I joined four tables to make sure that product name, orders and country were all available for query. I used the COUNT function to get number of orders as I assumed all orders would be treated as a single entity whether small or large. I ordered the count from highest to lowest and limited the output to 1 to give only the most frequently ordered product. Then I selected the Product name from the previous query.

```
SELECT ProductName
FROM(
	SELECT Products.ProductID, Products.ProductName, COUNT(*)
	FROM Orders
		LEFT JOIN Customers ON Orders.CustomerID = Customers.CustomerID
		LEFT JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
		LEFT JOIN Products ON OrderDetails.ProductID = Products.ProductID
	WHERE Country = 'Germany'
	GROUP BY 1
	ORDER BY 3 DESC
	LIMIT 1);
```