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!

Answers and comments are set to a **red** font color.

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.

   Currently, we are getting AOV by getting the average of the order amount column. AOV is an important metric because it tells us how much a customer spends each time they place an order. If we want to retain the core analysis from this metric, we should find the median of the order amount as opposed to the mean for this particular data set. This is because with the mean, there is outlier bias - that is, outliers can bias the mean and inflate it. For example, there are repeated orders with an order amount of $704,000 for 2000 items. This undoubtedly skews the data set and thus inflates the AOV. Instead, the median is unaffected by outliers, so the value would not be inflated. Effectively, we could still determine, in essence, how much a customer orders each time on “average”.

b. What metric would you report for this dataset?

   As noted above, I would determine the median order value for this data set. This effectively gives the same analysis as the average order value, only without outlier bias skewing the results.

c. What is its value?

Using this metric, we get $284 per order over the 30 day window.

The above was determined using an R program:

```
data <- read.csv('./2019 Winter Data Science
Intern Challenge Data Set - Sheet1.csv')
median(data$order_amount)
```

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?

   Going through the different tables that are available in the database, we see that the ones related to orders and shipping are **Orders** and **Shippers**. So, looking at the Orders table, we see there is a column, ShipperID, indicating who the Shipper is based on their ID. However, at the moment, we only know that we want orders shipped by Speedy Express, we don't know what the respective ShipperID is for them. This is where the Shippers table comes in. The Shippers table gives us a ShipperName and ShipperID key. It also gives the phone number, but that is irrelevant to us in this query. We see that the corresponding ShipperID for Speedy Express is 1. So, in essence, we are looking for the number of orders that have a ShipperID of 1.

   Now, how did I determine that the ShipperID of Speedy Express was 1? Well, the Shippers table only has 3 records, so simply taking a look, I was able to easily determine what the desired ShipperID was. However, suppose there we more than 3 records. Suppose there were so much that a simple check would not suffice. In a case like this, where we can't manually determine it, we could use a query to get the ShipperID that matches with the ShipperName desired:

   ```
   -- Speedy Express' ShipperID --
   SELECT ShipperId
   FROM [Shippers]
   WHERE ShipperName = 'Speedy Express'
   ```

   Now, this query is important because SQL doesn't have eyes of its own. So, we would need to help it determine what the ShipperID is with this query. You'll see what I mean soon.

Coming back to the ask of this question, we want to **count** the number of orders that have the ShipperID of 1, that is, the Speedy Express Shipper ID.

```sql
-- Number of Orders with ShipperID 1 --
SELECT COUNT(*)
FROM [Orders]
WHERE ShipperID = 1;
```

While technically this would work, it's better to foolproof this query. In the case that the Speedy Express Shipper ID were to change, this would no longer work. That's why, we need to combine the first and second query above, and finalize the query to be the following:

```sql
-- Number of Orders with Speedy Express' ShipperID --
SELECT COUNT(*)
FROM [Orders]
WHERE ShipperID = (
SELECT ShipperId
      FROM [Shippers]
      WHERE ShipperName = 'Speedy Express');
```

We then get a result of **54**, meaning that there were 54 orders shipped by Speedy Express in total.

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

For this question, we want to use tables that contain information on employees and their last name, as well as their orders. A quick look at the tables shows us that we are interested in **Employees** and **Orders**. We are interested in the Employees table because it provides us a list of employee last names and their respective EmployeeID. Why is the EmployeeID significant? Looking at the Orders table, we see that the information of employees is represented by their EmployeeID. Using the Orders table, we need to determine which EmployeeID has the most orders. Then, using the Employees table, we look up the last name that corresponds to this EmployeeID.

First, let's take a look at the EmployeeIDs and respective number of orders.

```sql
-- EmployeeID and their number of orders --
SELECT EmployeeID, COUNT(*)
FROM [Orders]
GROUP BY EmployeeID;
```

Next, since we're looking for the EmployeeID with the most orders, we should sort the results.

```
-- EmployeeID and their number of orders SORTED from highest
to lowest--
SELECT EmployeeID, COUNT(*)
FROM [Orders]
GROUP BY EmployeeID
ORDER BY COUNT(*) DESC;
```

Observing the results, we can see that the EmployeeID with the most Orders is 4. We can then go to the Employees table and find the LastName that corresponds with an EmployeeID of 4.

```
-- LastName of EmployeeID 4 --
SELECT LastName
FROM [Employees]
WHERE EmployeeID = 4;
```

Of course, the issue with this query is that, if the EmployeeID were to change, it would no longer be valid. So, we need to write our query to protect against that. Rather than hard coding in the desired EmployeeID, we can modify an earlier query to return the EmployeeID with the highest number of orders.

Let's revisit this query:

```
-- EmployeeID and their number of orders SORTED from highest
to lowest--
SELECT EmployeeID, COUNT(*)
FROM [Orders]
GROUP BY EmployeeID
ORDER BY COUNT(*) DESC;
```

Since we only want EmployeeID, we can remove the count from the select. We also only want the highest one. So, we can limit the result to only the first one. Our query then becomes:

```
-- EmployeeID with most orders--
SELECT EmployeeID
FROM [Orders]
```

```
GROUP BY EmployeeID
ORDER BY COUNT(*) DESC
LIMIT 1;
```

This will give us the EmployeeID with the most orders. So, we can then replace the hard coded value with this query. Our finalized query then becomes:

```
-- LastName of EmployeeID with most orders --
SELECT LastName
FROM [Employees]
WHERE EmployeeID = (
      SELECT EmployeeID
      FROM [Orders]
      GROUP BY EmployeeID
      ORDER BY COUNT(*) DESC
      LIMIT 1);
```

We then get a result of **Peacock**, meaning that the last name of the employee with the most orders is Peacock.

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

This query will be a much larger one than our previous two. We'll need to take it step-by-step.

First, we know from the question that we are interested in customers in Germany. We need to grab the CustomerID of those in Germany. Why the CustomerIDs? Because, if we look at the other tables, the only property related to the customer is via the CusomterID. So, we can do this using the following query:

```
-- CustomerID of those in Germany --
SELECT CustomerID
FROM [Customers]
WHERE Country = 'Germany';
```

Next, we know that we are interested in the products that were ordered. How do we determine what products were ordered? If we look at the Products table, there's a

ProductID. While handy, it’s not going to help us with just yet. There are a couple more steps before we get there. Looking at the other tables, there is an OrderDetails table that contains information on the Order and the ProductID along with the quantity. However, there’s nothing connecting the orders to the customer in this table, so we need to keep looking. With more searching, we’d actually find that the Orders table is what we’re actually interested in. This table connects the CustomerID with the OrderID. So now, what we’re looking for are the OrderIDs of the CustomerIDs found in the earlier query - that is, the CustomerIDs of the German customers.

```
-- OrderIDs of Customers with CustomerIDs associated with
Customers in Germany --
SELECT CustomerID, OrderID
FROM [Orders]
WHERE CustomerID IN (
    SELECT CustomerID
    FROM [Customers]
    WHERE Country = 'Germany')
ORDER BY CustomerID
```

We actually don’t need to order by CustomerID, but I did just to make looking through the results a bit easier.

Now, with these IDs, we can determine which products were purchased along with the quantity purchased.

```
 -- Get ProductID and Quantity Purchased by German Customers --
SELECT ProductID, SUM(Quantity)
FROM [OrderDetails]
WHERE OrderID IN (
    SELECT OrderID
    FROM [Orders]
    WHERE CustomerID
    IN (
        SELECT CustomerID
        FROM [Customers]
        WHERE Country = 'Germany'))
GROUP BY ProductID
ORDER BY SUM(Quantity) DESC;
```

We can then modify this query to show only the first result and only the ProductID. This ProductID represents the product with the most orders (or quantity bought) by German customers.

```sql
-- Get ProductID of most purchased product --
SELECT ProductID
FROM [OrderDetails]
WHERE OrderID IN (
    SELECT OrderID
    FROM [Orders]
    WHERE CustomerID IN (
        SELECT CustomerID
        FROM [Customers]
        WHERE Country = 'Germany'))
GROUP BY ProductID
ORDER BY SUM(Quantity) DESC
LIMIT 1;
```

We then use the ProductID we got above to find the name of the product.

```sql
-- Get product name with associated product id --
SELECT ProductName
FROM [Products]
WHERE ProductID = (
    SELECT ProductID
    FROM [OrderDetails]
    WHERE OrderID IN (
        SELECT OrderID
        FROM [Orders]
        WHERE CustomerID IN (
            SELECT CustomerID
            FROM [Customers]
            WHERE Country = 'Germany'))
    GROUP BY ProductID
    ORDER BY SUM(Quantity) DESC
    LIMIT 1);
```

We get a result of **Boston Crab Meat**, meaning that it was the most ordered product by customers in Germany.