Winter 2021 Data Science Intern Challenge

Doina Covaliu

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.

Current calculation:

AOV =Total revenue / number of orders = $3145.13

```
high_value_orders = df.groupby(['total_items',"order_amount"])\
                        .size().reset_index(name='Number of Orders')\
                        .sort_values(by='order_amount', ascending=False)
high_value_orders.head(5)
```

	total_items	order_amount	Number of Orders
275	2000	704000	17
273	6	154350	1
230	4	102900	1
173	3	77175	9
115	2	51450	16

```
df.order_amount.describe()
```

```
count      5000.000000
mean       3145.128000
std       41282.539349
min          90.000000
25%         163.000000
50%         284.000000
75%         390.000000
max      704000.000000
Name: order_amount, dtype: float64
```

In the current methodology, the **AOV** is calculated as the **mean** of all the orders amounts. In this approach, the mean is not an accurate representation of the AOV as the dataset has a high standard deviation of 41,282.53 that indicates that the values are spread over a wide range, between $90 and $704,000 per order.

The **AOV** is misleading as few extremely high order amounts skew the results.

Solution:

The dataset seems to contain information regarding 2 types of customers:

- **B2B (business to business)** that order more than 10 items per order.
- **D2C** (**direct to customer)** that order up to 10 items per order.

A better way to evaluate the data is to separate the dataset into B2B and D2C subsets and calculate the AOV for each one of them separately. Please see below the proposed code solution.

```
import sys
import pandas as pd

def main():

        data =  pd.read_csv('orders.csv', header=0)
        df = pd.DataFrame(data)

        #Question 1.a.Separate the dataset in D2C AND B2C
        #D2C -customer who order up to 10 items per order
        #B2B-customers who order more than 10 items per order

        D2C=df[df.total_items <= 10]
        AOV_D2C= D2C.order_amount.mean()
        print('Question 1.a: \n Average Order Value for D2C customers is: $',AOV_D2C)

        B2B=df[df.total_items >10]
        AOV_B2B= B2B.order_amount.mean()
        print(' Average Order Value for B2B customers is: $',AOV_B2B)

        #Question 2.b and c Calculate the Median Order Value for the entire dataset

        Median_Order_Value= df.order_amount.median()
        print('Question 1.b and 1.c: \n Median Order Value is: $',Median_Order_Value)

        #Average basket value for B2B customers
        ABV_B2B= B2B.total_items.mean()
        print(" Average basket value for B2B customers is:",ABV_B2B,'items' )

        #Average basket value for D2C customers
        ABV_D2C= D2C.total_items.mean()
        print(" Average basket value for D2C customers is:",ABV_D2C,'items' )

if __name__ == "__main__":
        main()
```

b. **What metric would you report for this dataset?**

Solution:

The **Median Order Value** is a better way to determine how much customers typically pay per order, if the entire dataset is being analyzed, rather than splitting it into D2C and B2B customers. The **Median Order Value** eliminates the outliers at the higher end of the order amounts since it is simply the middle value.

The **Average Basket Value** is also a very useful metric that shows how many items are typically sold per transaction. In this dataset, different shops sell the same product at different prices from $90 to $25,725 per item. Considering the wide price range and the different approach for marketing the same product as either a basic clothing item or a luxury item, **the average basket** value is a better way to determine the size of the average order compared to the average order value.

c. **What is its value?**

Solution:

- Median Order Value =$284
- Average Basket Value for B2B customers is **2000** items per order
- Average Basket Value for D2C customers is **1.99** items per order (approximately 2 items per order)

SQL Challenge:

Q 2.a: How many orders were shipped by Speedy Express in total?

I. A JOIN clause will combine the **Orders** and **Shippers** tables, based on the common column **ShippersID**.
II. The new table resulted from the join operation will be filtered to include only those records where the **Shipper Name** is **Speedy Express**.
III. The resulting records will be counted and displayed. The number represents the total number of orders shipped by **Speedy Express**.

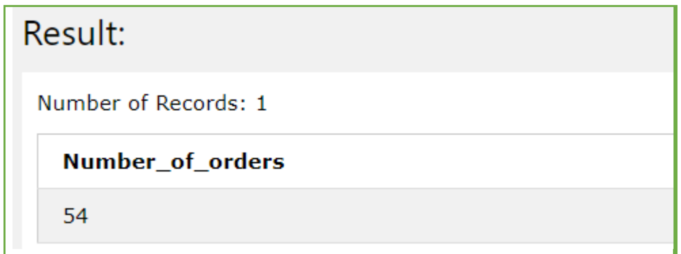
The number of orders shipped by Speedy Express is 54.

```
SELECT count(*) as Number_of_orders

FROM Shippers JOIN Orders

ON Shippers.ShipperID= Orders.ShipperID

WHERE ShipperName="Speedy Express";
```



Result:

Number of Records: 1

Number_of_orders
54

Q 2. b What is the last name of the employee with the most orders?

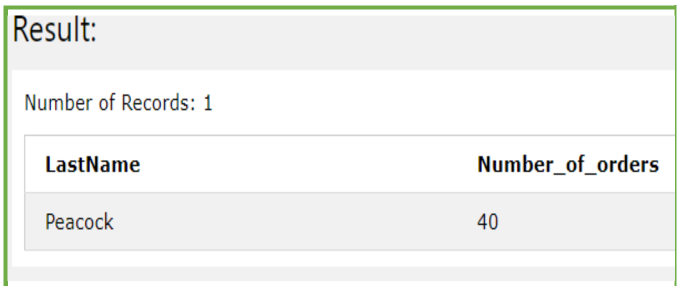
I. A join operation will combine **Employees** and **Orders** tables based on the common column **EmployeeID**.

II. From the newly generated table after the join operation, the rows that have the same values in the **LastName** column will be grouped (summarized). In other words, we calculate the total number of orders for each **Last Name**.

III. Each record containing the last name and the corresponding total number of orders will be arranged in descending order.

IV. The last name with the most orders will be at the top. Only the first record will be displayed which is the last name of the employee with most orders.

```
SELECT LastName, count(*) as Number_of_orders
FROM Employees JOIN Orders
ON Employees.EmployeeID=Orders.EmployeeID
GROUP BY LastName
ORDER BY Number_of_orders DESC limit 1;
```



Result:

Number of Records: 1

LastName	Number_of_orders
Peacock	40

Q 2.c What product was ordered the most by customers in Germany?

The information necessary to answer this question is located in multiple tables: **Orders, Customers, Products, and OrderDetails**. To answer this question 3 join operations will be performed:

I. In the first join operation, the **Orders** and **Customers** tables will be combined based on the common column **OrderID**. From the resulting table, the **OrderID** of the customers from Germany will be extracted.

II. Through the second join clause, **Products** and **OrderDetails** tables will be combined based on **ProductId** column. This time, the **ProductName, Quantity**, and the corresponding **OrderID** will be extracted.

III. The result of the first 2 join operations will be combined in a new table using the **OrderID** to make a connection between the results of the first 2 joins. Then, we extract the **ProductName** and **Quantity** (number of items in the order).

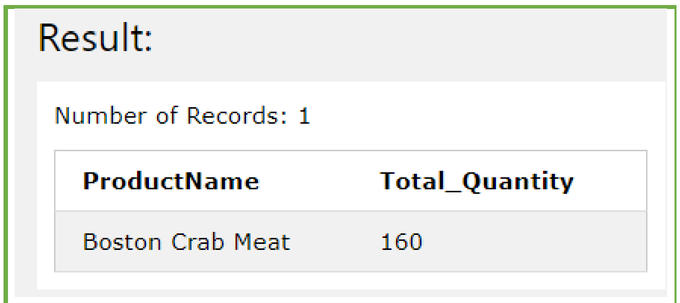
IV. In the next step, the rows that have the same values in the **ProductName** column will be grouped (summarized). In other words, we calculate the total number of items (**Quantity**) for each **Product Name**.

V. Afterward, the resulting records will be organized in descending order. The **ProductName** with the highest quantity will be on top and it is the only one that will be displayed.

VI. That represents the **product that was ordered the most by customers in Germany:**

Solution: Boston Crab Meat with a quantity of 160 items

```
SELECT ProductName, Sum(b.Quantity) as
Total_Quantity
FROM
            (
             SELECT OrderID
            FROM Customers JOIN Orders
            ON  Customers.CustomerID=Orders.CustomerID
            WHERE Country="Germany"
            ) AS a
 JOIN
            (
            SELECT OrderID, ProductName, Quantity
            FROM Products JOIN OrderDetails
            ON Products.ProductID=OrderDetails.ProductID
            ) AS b
ON a.OrderID=b.OrderID
GROUP BY ProductName
ORDER BY Total_Quantity DESC limit 1;
```



Result:

Number of Records: 1

ProductName	Total_Quantity
Boston Crab Meat	160