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.
b. What metric would you report for this dataset?
c. What is its value?

jupyter Summer 2022 Data Science Intern Challenge Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

In [1]:
```
# Libraries used
import pandas as pd     # Pandas library
```

In [2]:
```
# Reading the csv file
file = pd.read_csv("2019 Winter Data Science Intern Challenge Data Set - Sheet1.csv", index_col = 'order_id')
file.head()
```

Out[2]:

order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
1	53	746	224	2	cash	2017-03-13 12:36:56
2	92	925	90	1	cash	2017-03-03 17:38:52
3	44	861	144	1	cash	2017-03-14 4:23:56
4	18	935	156	1	credit_card	2017-03-26 12:43:37
5	18	883	156	1	credit_card	2017-03-01 4:35:11

In [3]:
```
# Data Frame dimension
file.shape
```

Out[3]: (5000, 6)

In [4]:
```
# Number of Sneakers shops
shops = len(pd.unique(file['shop_id']))
shops
```

Out[4]: 100

In [5]:
```
# Number of users
user = len(pd.unique(file['user_id']))
user
```

Out[5]: 301

In [6]:
```
file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 1 to 5000
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   shop_id         5000 non-null   int64
 1   user_id         5000 non-null   int64
 2   order_amount    5000 non-null   int64
 3   total_items     5000 non-null   int64
 4   payment_method  5000 non-null   object
 5   created_at      5000 non-null   object
dtypes: int64(4), object(2)
memory usage: 273.4+ KB
```

In [7]:
```
# Converting the column data type
file['created_at'] = pd.to_datetime(file['created_at'])
file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 1 to 5000
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   shop_id         5000 non-null   int64
 1   user_id         5000 non-null   int64
 2   order_amount    5000 non-null   int64
 3   total_items     5000 non-null   int64
 4   payment_method  5000 non-null   object
 5   created_at      5000 non-null   datetime64[ns]
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 273.4+ KB
```

In [8]:
```
# Checking the Day window of the dataset
file.sort_values(by ='created_at')
```

Out[8]:

order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
1863	39	738	536	4	cash	2017-03-01 00:08:09
1742	39	910	268	2	cash	2017-03-01 00:10:19
3229	97	912	324	2	cash	2017-03-01 00:14:12
1268	80	798	290	2	credit_card	2017-03-01 00:19:31
2690	49	799	258	2	credit_card	2017-03-01 00:22:25
...	...	...	...	...	...	...
2631	53	940	112	1	credit_card	2017-03-30 23:12:13
1686	34	818	244	2	cash	2017-03-30 23:16:10
1475	21	815	142	1	cash	2017-03-30 23:26:54
318	52	848	292	2	cash	2017-03-30 23:41:34
2458	95	700	168	1	credit_card	2017-03-30 23:55:35

5000 rows × 6 columns

In [9]:
```
# Naively Calculated average order value.
n_cal = round(sum(file['order_amount'])/file.shape[0] , 2)
print("Naively calculated average order value (AOV): $%s" % n_cal)
```

```
Naively calculated average order value (AOV): $3145.13
```

The figure of 3145.13 dollar is calculated by dividing the 'Total Order Amount' with the 'Number of Orders'. While calculation, we ignore the fact that in single order the customer could have purchased more than one item. This misjudgement led to wrong average order value.

Instead of 'Number of Orders' we should have used the 'Total Number of Items' purchased by the customers and divided from 'Total Order Amount'. By making changes we get 357.92 dollars of Average Order Value which is better way to evaluate this data.

In [10]:
```
# 'Total Order Amount (in 30 days)' divided by 'Total Number of Items' purchased in 30 days
a_cal = round(sum(file['order_amount'])/sum(file['total_items']),2)
print("Actual average order value (AOV): $%s"% a_cal)
```

```
Actual average order value (AOV): $357.92
```

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?

SQL Statement:

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

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 1

COUNT(Shippers.ShipperName)
54

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Restore Database

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

SQL Statement:

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

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 1

LastName	NumberOfOrders
Peacock	40

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

1. Table ‘Products_orderdetails’ created by joining tables ‘OrderDetails’ and ‘Products’

SQL Statement:

```
CREATE TABLE Products_orderdetails AS
SELECT OrderDetails.OrderID, OrderDetails.ProductID, Products.ProductName, OrderDetails.Quantity
FROM OrderDetails
LEFT JOIN Products ON OrderDetails.ProductID = Products.ProductID;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

You have made changes to the database.

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29
Products_orderdetails	518

Restore Database

2. Table ‘Cust_orders’ created by joining tables ‘Orders’ and ‘Customers’

SQL Statement:

```
CREATE TABLE Cust_orders AS
SELECT Orders.OrderID, Customers.CustomerID, Customers.Country
FROM Orders
LEFT JOIN Customers ON Orders.CustomerID = Customers.CustomerID
WHERE Country = 'Germany';
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

You have made changes to the database.

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29
Products_orderdetails	518
Cust_orders	25

Restore Database

3. The product ordered the most by customers in Germany: **Gorgonzola Telino**

SQL Statement:

```
SELECT Products_orderdetails.ProductName, COUNT(Products_orderdetails.OrderID) AS NumberOfOrders
FROM Cust_orders
LEFT JOIN Products_orderdetails ON Cust_orders.OrderID = Products_orderdetails.OrderID
GROUP BY ProductName
ORDER BY NumberOfOrders DESC LIMIT 1;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 1

ProductName	NumberOfOrders
Gorgonzola Telino	5

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29
Products_orderdetails	518
Cust_orders	25

Restore Database