Walmart Data Visualization & Wrangling - Project

12.05.2023

—

Company: Walmart
Hala Basim Sedki
Reem Abdeghany Mohamed
Ebraam Hani Attia

Data Source

According to many professors and courses that we have taken at ESLSCA University, many professors have mentioned that Walmart has several times put their data online to challenge people from across the world to drive some insights and analytics from them. We took it as a challenge to use real-world data and gather our own insight for this project.

Goals

1. Dashboard Creation: Identify the KPIs, design an intuitive and visually appealing dashboard, add interactive visualizations and filtering capabilities to allow users to explore the data at various levels of granularity
2. Data Analysis: Provide valuable insights to business entities regarding the effectiveness of their sales strategies through visualization and charts
3. Sales Forecasting: Leverage historic data and apply time serie generate sales forecasts for next 15 days
4. Actionable Insights and Recommendations: End goal is to provide insights and actionable information that can drive strategic decision support the supermarket's goals for growth, efficiency, satisfaction.

Data Preparation - Wrangling

The data was gathered from an official source on Kaggle. After exploring it for a while, we needed to check that the data is 100% clean and ready to use to ensure accurate insights. From a look before cleaning, it seemed like there were no missing values except for one column - the Postal Code for a specific city. This made us sure that we needed to start the data cleaning process.

Tools: Jupyter Notebook, SQL

I. Data Exploration

Using a jupyter notebook (in python language), we started exploring our data. Here is a code snippet of what we had:

Exploring the Data

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn import pipeline
        from sklearn.preprocessing import LabelEncoder
        from scipy import stats
        import time
```

```
In [2]: walmart_df = pd.read_csv('walmart_data.csv')
```

```
In [3]: walmart_df.head()
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Ca
0	1	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-BO-10001798	Furniture	Boo
1	2	CA-2017-152156	08/11/2017	11/11/2017	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420.0	South	FUR-CH-10000454	Furniture	
2	3	CA-2017-138688	12/06/2017	16/06/2017	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036.0	West	OFF-LA-10000240	Office Supplies	
3	4	US-2016-108966	11/10/2016	18/10/2016	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311.0	South	FUR-TA-10000577	Furniture	

As seen above, we imported many libraries that we were going to use throughout the data exploration process. From the looks of the head, it looks like we needed to change the data type for the postal code to be an integer.

```
In [4]: walmart_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9800 entries, 0 to 9799
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9800 non-null   int64
 1   Order ID       9800 non-null   object
 2   Order Date     9800 non-null   object
 3   Ship Date      9800 non-null   object
 4   Ship Mode      9800 non-null   object
 5   Customer ID    9800 non-null   object
 6   Customer Name  9800 non-null   object
 7   Segment        9800 non-null   object
 8   Country        9800 non-null   object
 9   City           9800 non-null   object
 10  State          9800 non-null   object
 11  Postal Code    9789 non-null   float64
 12  Region         9800 non-null   object
 13  Product ID     9800 non-null   object
 14  Category       9800 non-null   object
 15  Sub-Category   9800 non-null   object
 16  Product Name   9800 non-null   object
 17  Sales          9800 non-null   float64
dtypes: float64(2), int64(1), object(15)
memory usage: 1.3+ MB
```

II. Data Analysis

A. Checking for Data Types:

As seen in the picture to the left, we started seeing the data type for each of our columns. As mentioned before, it was found that we need to change the data type for:

- Postal Code
- Ship Date
- Order Date

B. Check for Missing Values

As seen below, we started coding to check for missing values in the data. It was found in fact as we hypothesized that there were missing values for postal code. We then further analyzed to find that all the missing data belonged to one specific city Burlington, Vermont, that was missing it's postal code.

Check for Missing Values

```
In [8]: walmart_df.loc[walmart_df["Postal Code"].isna(),['Country','City','State','Postal Code']]
```

Out[8]:

	Country	City	State	Postal Code
2234	United States	Burlington	Vermont	NaN
5274	United States	Burlington	Vermont	NaN
8798	United States	Burlington	Vermont	NaN
9146	United States	Burlington	Vermont	NaN
9147	United States	Burlington	Vermont	NaN
9148	United States	Burlington	Vermont	NaN
9386	United States	Burlington	Vermont	NaN
9387	United States	Burlington	Vermont	NaN
9388	United States	Burlington	Vermont	NaN
9389	United States	Burlington	Vermont	NaN
9741	United States	Burlington	Vermont	NaN

```
In [9]: walmart_df.loc[(walmart_df['City']=='Burlington') & (walmart_df['State']=='Vermont'),['Country','Postal Code']]
```

Out[9]:

	Country	Postal Code
2234	United States	NaN
5274	United States	NaN
8798	United States	NaN
9146	United States	NaN
9147	United States	NaN
9148	United States	NaN
9386	United States	NaN

To correct this for our clean dataset, I explored outside what the postal code for Burlington was which is 05401. Then using the fill null function (fillna), we put 05401. This made us eliminate the challenge of the missing data. In order to be 100% sure that there is no more missing data, we made the python program check if there are any further null values to which there was 0 (as shown below):

```
In [12]: train = walmart_df.copy()
         train['Postal Code'] = train['Postal Code'].fillna(5401) # leading zeros in decimal integer literals are not permitt
```

I checked the code for the city of Burlington as it was the one with missing values and found the postal code as 05401, this is why in the fillna, I assigned the postal code as 5401

```
In [13]: train.isna().sum().sum()
Out[13]: 0
```

C. Check for Duplicate Data:

We needed to further check if there is any duplicated data, as this is crucial for the data integrity and cleanliness.

This will allow us to ensure that all the insights are accurate.

Check for Duplicate Data

In [15]:
```
train[train.duplicated()]
```

Out[15]:

	Order Date	Ship Date	Ship Mode	Customer ID	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Product Name	Sales
3406	23/04/2015	27/04/2015	Standard Class	LB-16795	Home Office	United States	Columbus	Ohio	43229.0	East	Furniture	Chairs	Global Leather Highback Executive Chair with P...	281.372

In [16]:
```
train.drop_duplicates(inplace=True)
train.duplicated().sum()
```

Out[16]: 0

As seen above, there was only one piece of duplicated data, which were found in rows 3405 and 3406. This meant that we further had to remove any duplicates, in which we did.

D. Change the Data Types:

As mentioned before, we needed to change the data types for three columns: postal code, ship date, and order date, since this would pose us with a challenge once the data is put into powerbi, since it will not accurately detect the dates.

For the Order Date & Ship Date:

Convert Data Types

In [17]:
```
train['Order Date'] = pd.to_datetime(train['Order Date'], format='%d/%m/%Y')
train['Ship Date'] = pd.to_datetime(train['Ship Date'], format='%d/%m/%Y')
train['Postal Code'] = train['Postal Code'].astype(int)
```

The above was done to format the date correctly in order to later visualize it correctly, so we assigned it the datetime data type

In [18]:
```
train.insert(loc=4,  column='order_month_year',value=train['Order Date'].dt.to_period('M'))
train.insert(loc=5, column='ship_month_year', value=train['Ship Date'].dt.to_period('M'))

train.insert(loc=6, column='order_day', value=train['Order Date'].dt.day)
train.insert(loc=7, column='order_month', value=train['Order Date'].dt.month)
train.insert(loc=8, column='order_year', value=train['Order Date'].dt.year)

train.insert(loc=9, column='ship_day', value=train['Ship Date'].dt.day)
train.insert(loc=10, column='ship_month', value=train['Ship Date'].dt.month)
train.insert(loc=11, column='ship_year', value=train['Ship Date'].dt.year)
```

Here the same was done but we seperated the month & year

To check that the data types are all correct:

As seen below, the data types are now all correct and in the format that they should be which means that we can now proceed with visualizing our data on PowerBI.

```
In [19]: train.info()

<class 'pandas.core.frame.DataFrame'>
Index: 9799 entries, 0 to 9799
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Order Date        9799 non-null   datetime64[ns]
 1   Ship Date         9799 non-null   datetime64[ns]
 2   Ship Mode         9799 non-null   object
 3   Customer ID       9799 non-null   object
 4   order_month_year  9799 non-null   period[M]
 5   ship_month_year   9799 non-null   period[M]
 6   order_day         9799 non-null   int32
 7   order_month       9799 non-null   int32
 8   order_year        9799 non-null   int32
 9   ship_day          9799 non-null   int32
 10  ship_month        9799 non-null   int32
 11  ship_year         9799 non-null   int32
 12  Segment           9799 non-null   object
 13  Country           9799 non-null   object
 14  City              9799 non-null   object
 15  State             9799 non-null   object
 16  Postal Code       9799 non-null   int64
 17  Region            9799 non-null   object
 18  Category          9799 non-null   object
 19  Sub-Category      9799 non-null   object
 20  Product Name      9799 non-null   object
 21  Sales             9799 non-null   float64
```

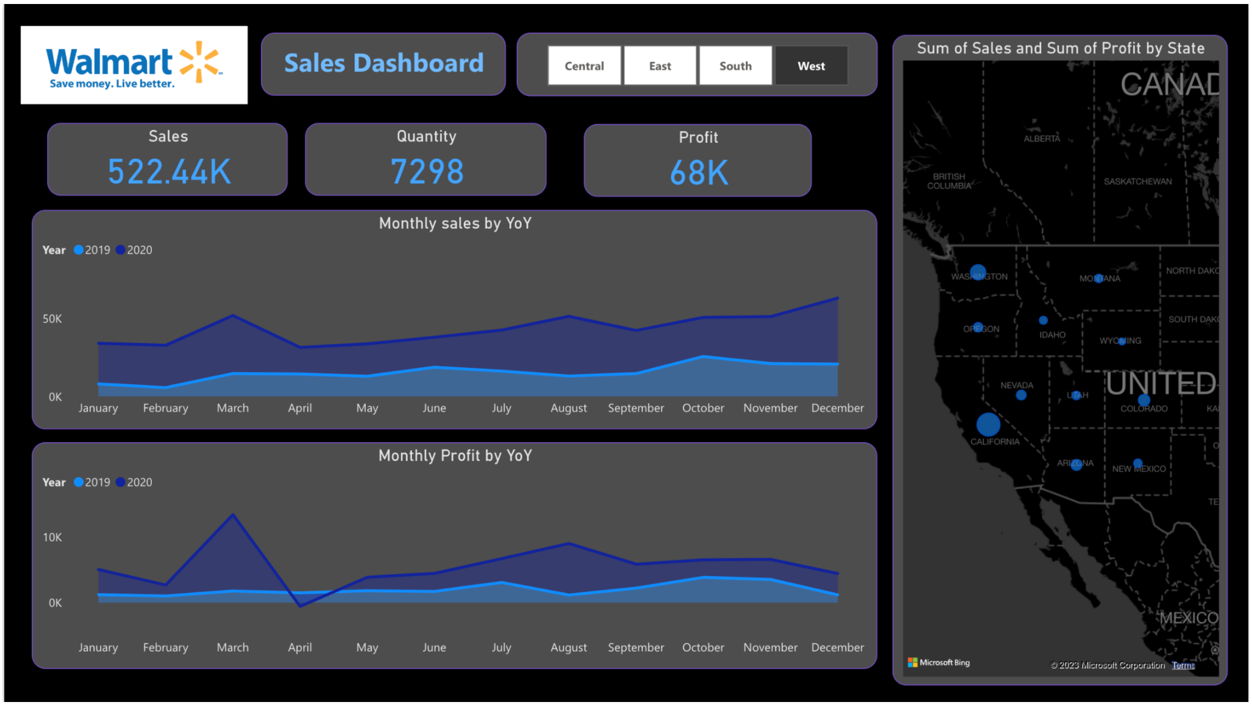
Data Visualization - Power BI

Tools: PowerBI

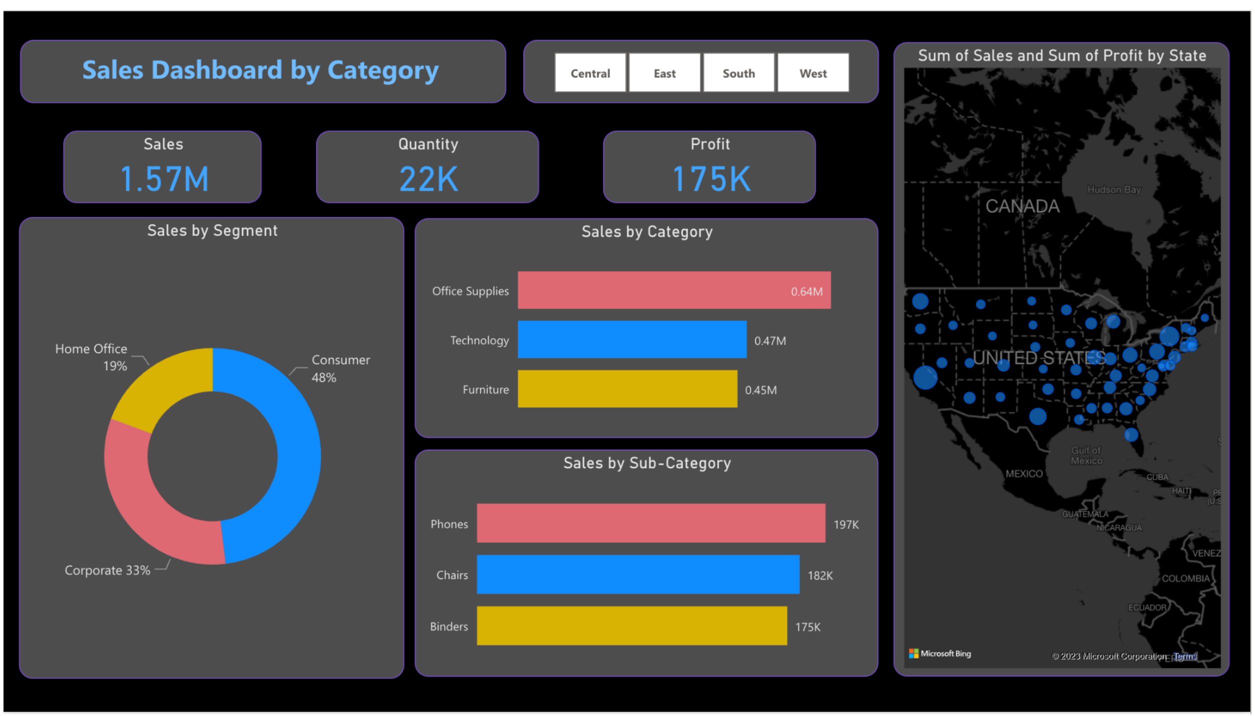
Steps:

1. Created a shared workspace for the 3 team members
2. Import the dataset from the csv file we had newly saved
3. Created New Functions such as:
    a. AvgDelivery: Finds the average time it takes to deliver an order.
    b. Sum of Profit: Calculated total profit.
    c. Sum of Sales: Calculated the number of sales conducted.
    d. Sum of Quantity:
    e. Started Visualizing

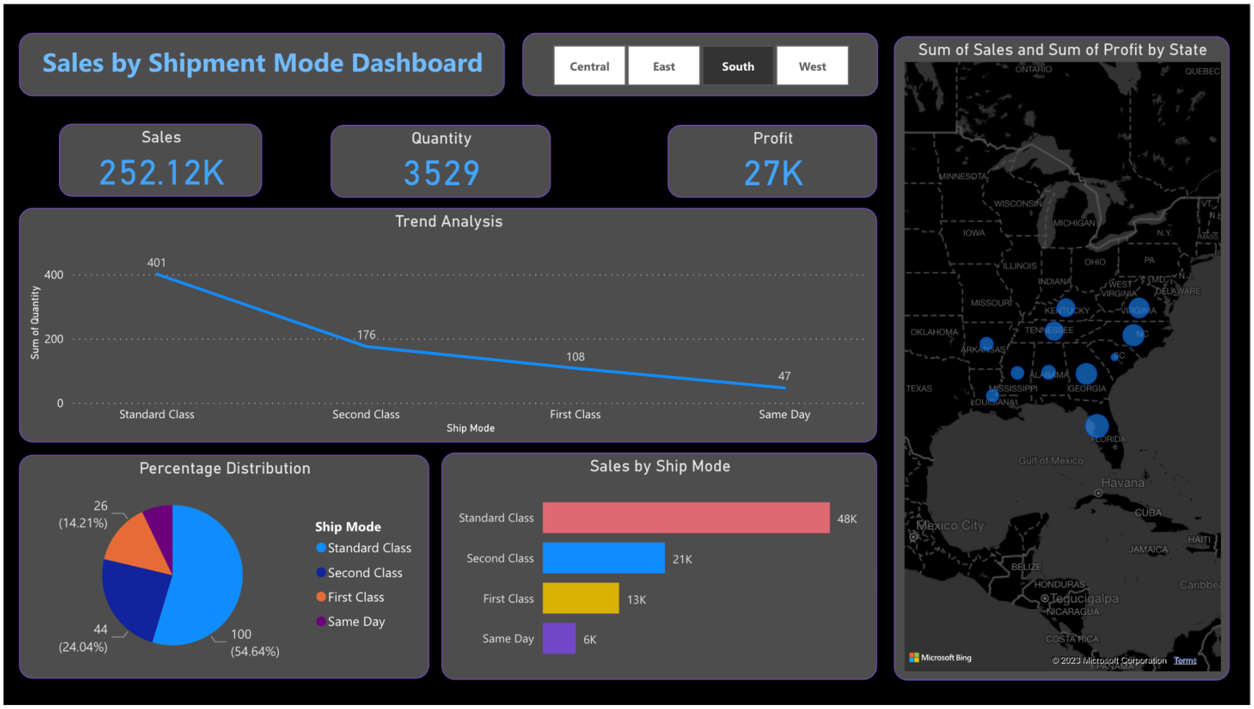
I. Sales Dashboard:



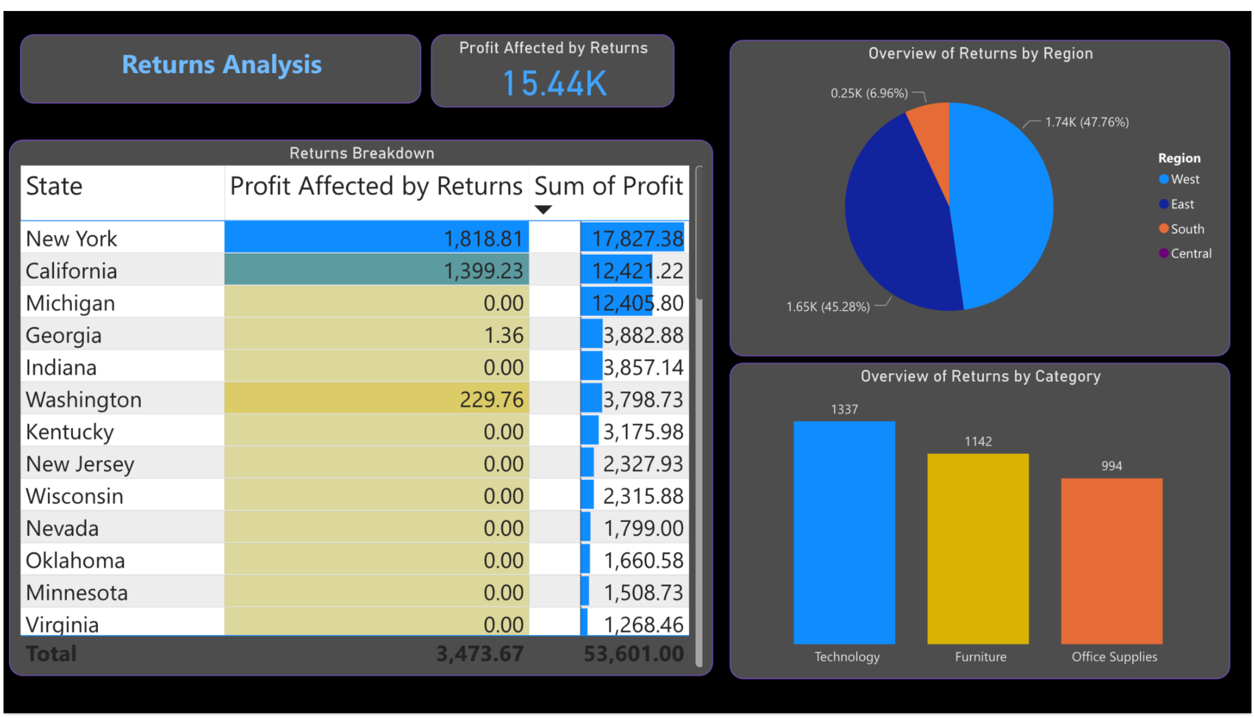
II. Sales Dashboard by Category:



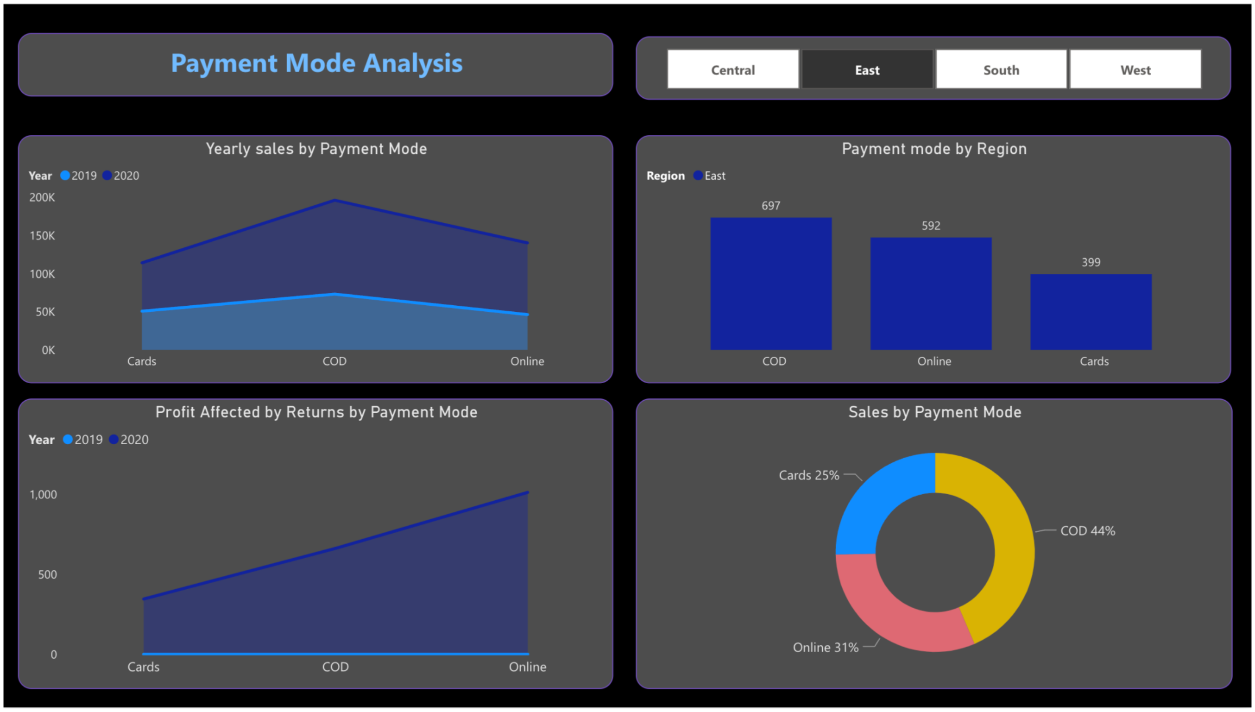
III. Shipment Method:



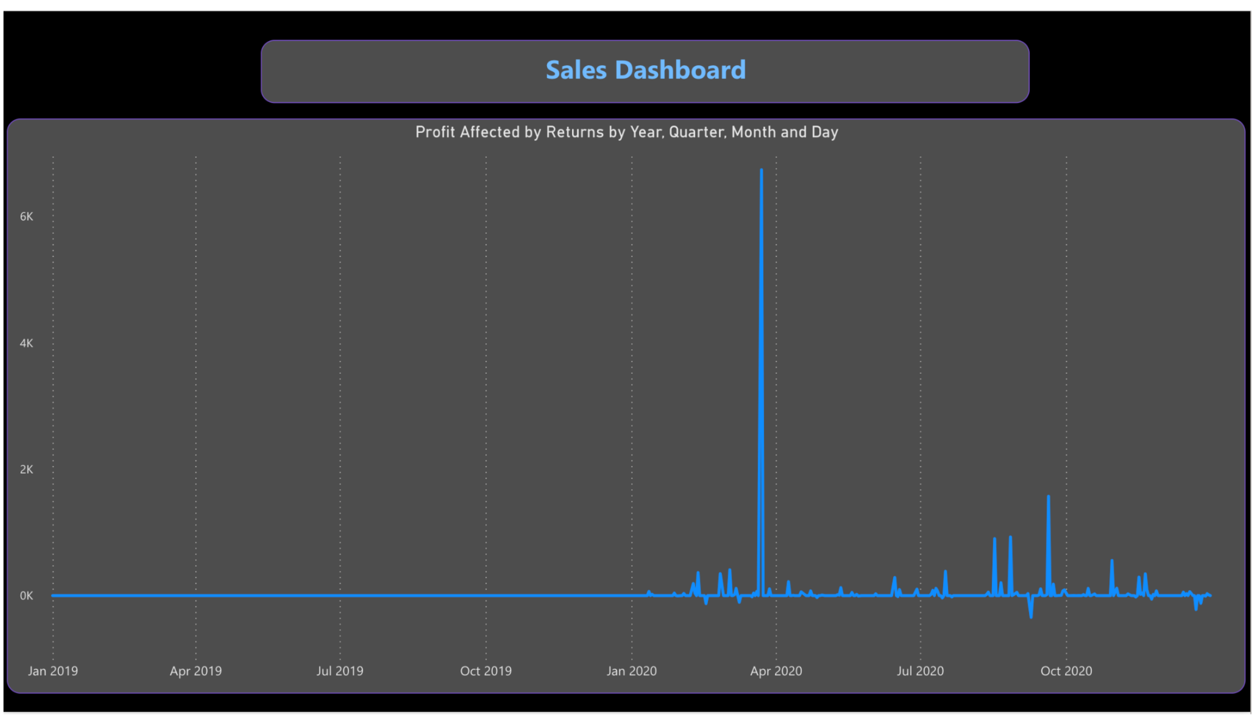
IV. Returns Analysis:



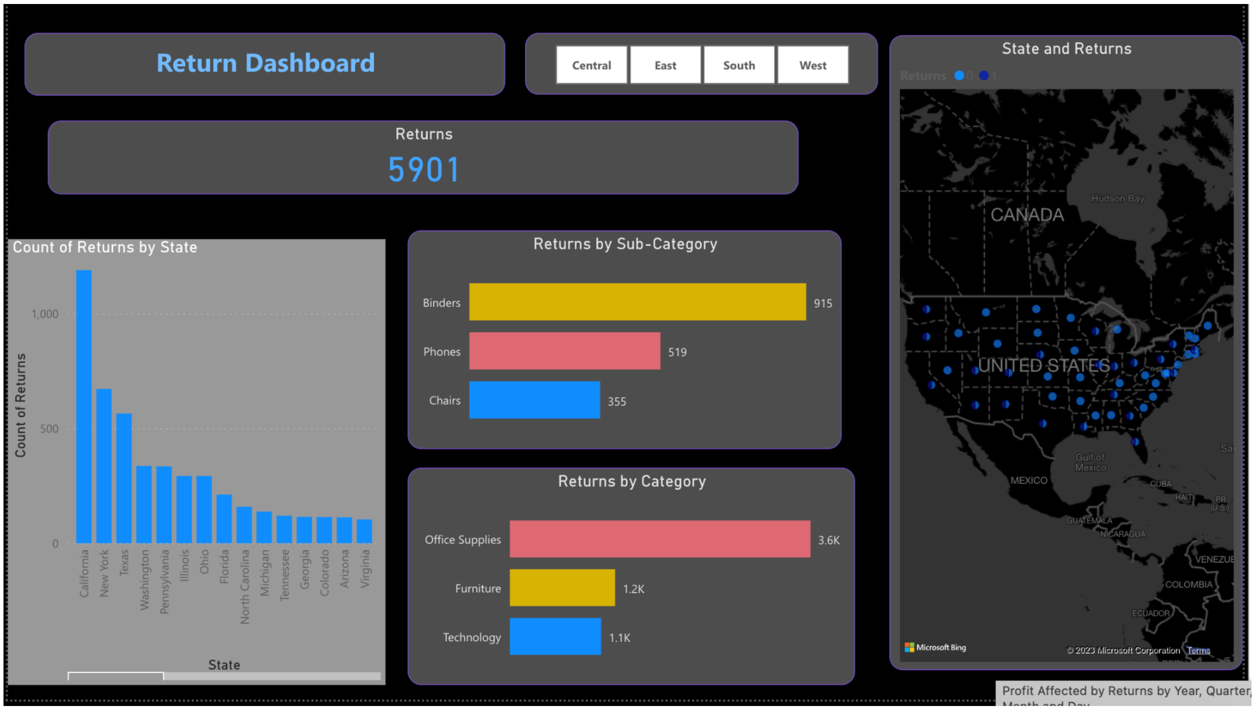
V. Payment Methods Analysis:



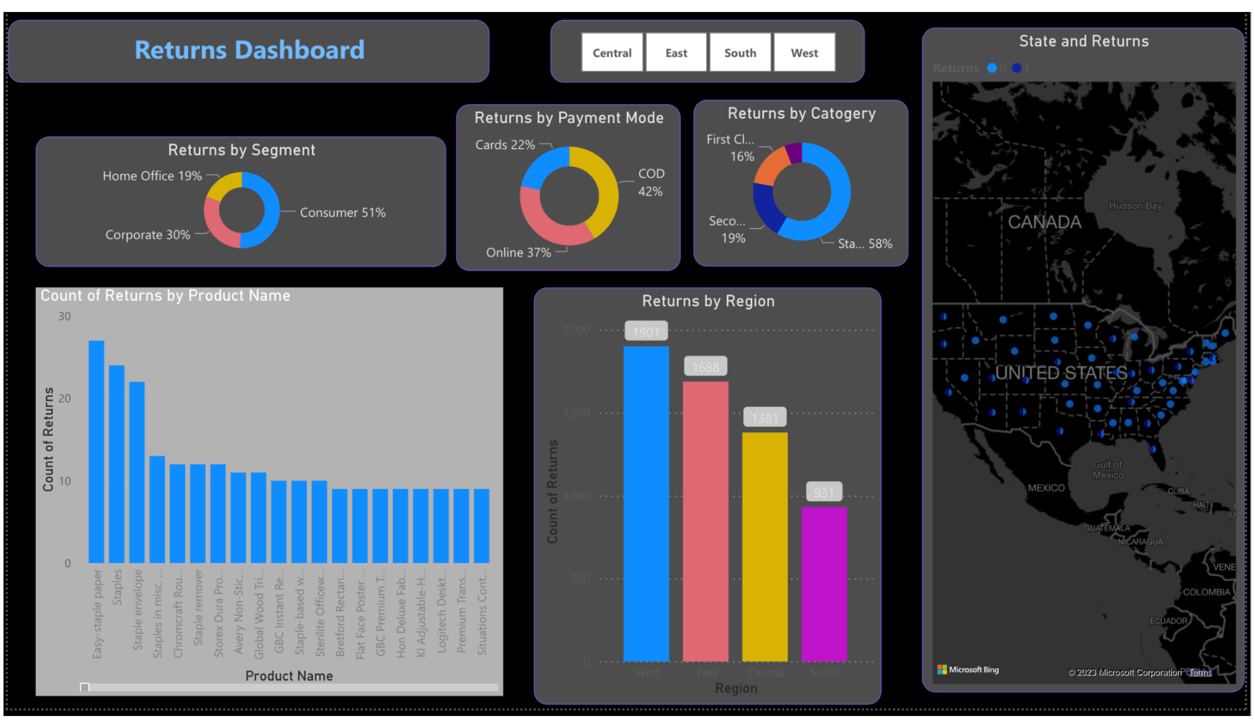
VI. Sales Affected by Returns Analysis:



VII. Returns Analysis 1:



VIII. Returns Analysis 2:



Useful Insights

Maximum sales are driven through COD payment mode.

Maximum sales are from the Customer segment (48.09%) and then corporate(32.55%).

Office supplies is the category that has the maximum sales.

Most of the customers preferred standard class ship mode.

Next 15 Days Forecast which is very useful business.

Maximum sales happened in the west region.

Maximum Profit earn in the month of October & December.

Average taking 4 days to ship the products.

Highest no of sales happened in the month of September, November & December

State-wise Maximum number of sales happened in California