Analyzing Amazon Sales Data

Wireframe Documentation

Revision Number - 1.0

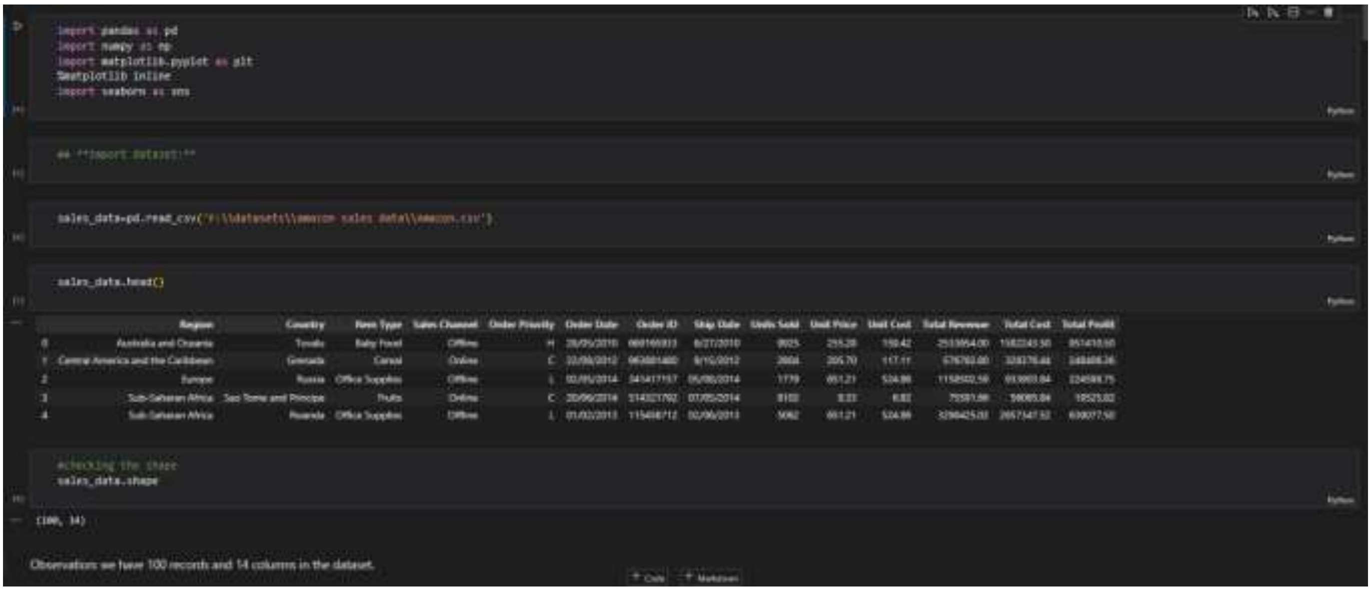
Last Date of Revision - 14/04/2023

Mohammad Aadil

Performed Exploratory Data Analysis :-

Exploratory Data Analysis (EDA) is an approach to analyzing data that emphasizes gaining an understanding of the data before formal modeling or hypothesis testing is carried out. EDA is a crucial step in the data analysis process, as it can help identify patterns, anomalies, and relationships between variables, and provide insights into the underlying structure of the data

EDA typically involves a combination of visual and numerical methods to explore the data. Visual methods include techniques such as scatterplots, boxplots, histograms, and heatmaps, which allow us to visualize the distribution of variables and the relationships between them. Numerical methods include summary statistics such as mean, median, and standard deviation, as well as correlation coefficients and regression analysis.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
## **Import dataset:**
```

```
sales_data=pd.read_csv('F:\\datasets\\amazon sales data\\amazon.csv')
```

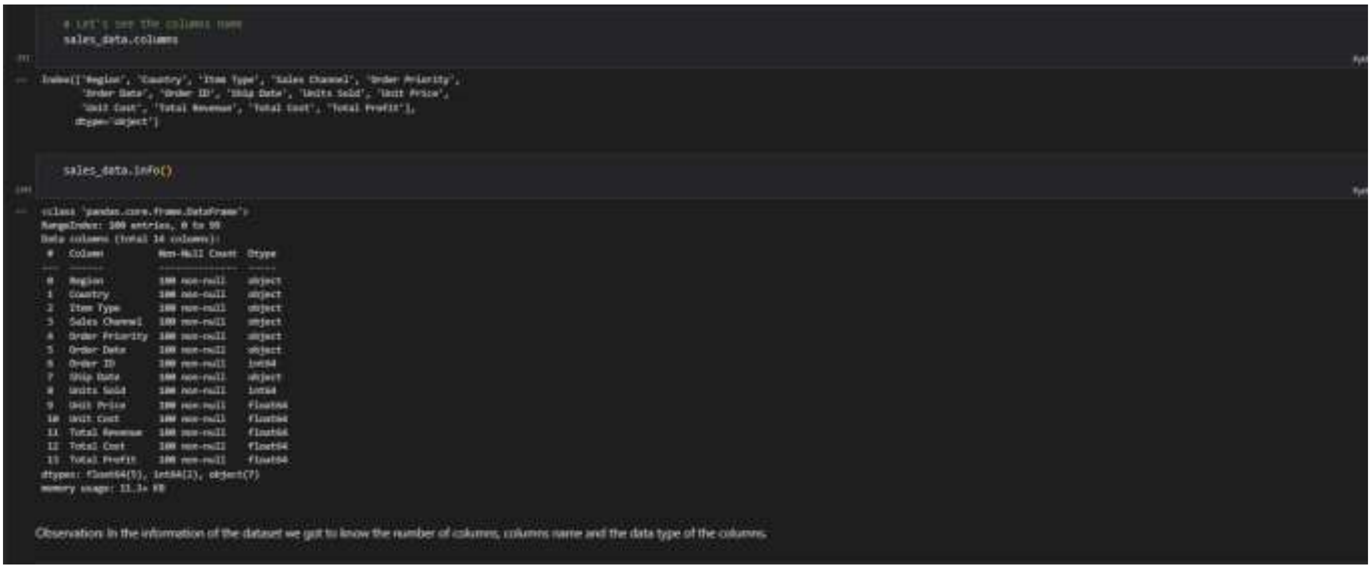
```
sales_data.head()
```

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	28/05/2010	669165933	6/27/2010	9925	255.28	159.42	2533654.00	1582243.50	951410.50
1	Central America and the Caribbean	Grenada	Cereal	Online	C	22/08/2012	963881480	9/15/2012	2804	205.70	117.11	576782.80	328376.44	248406.36
2	Europe	Russia	Office Supplies	Offline	L	02/05/2014	341417157	05/08/2014	1779	651.21	524.96	1158502.59	933903.84	224598.75
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	20/06/2014	514321792	07/05/2014	8102	9.33	6.92	75591.66	56065.84	19525.82
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	01/02/2013	115456712	02/06/2013	5062	651.21	524.96	3296425.02	2657347.52	639077.50

```
#checking the shape
sales_data.shape
```

```
(100, 14)
```

Observation: we have 100 records and 14 columns in the dataset.



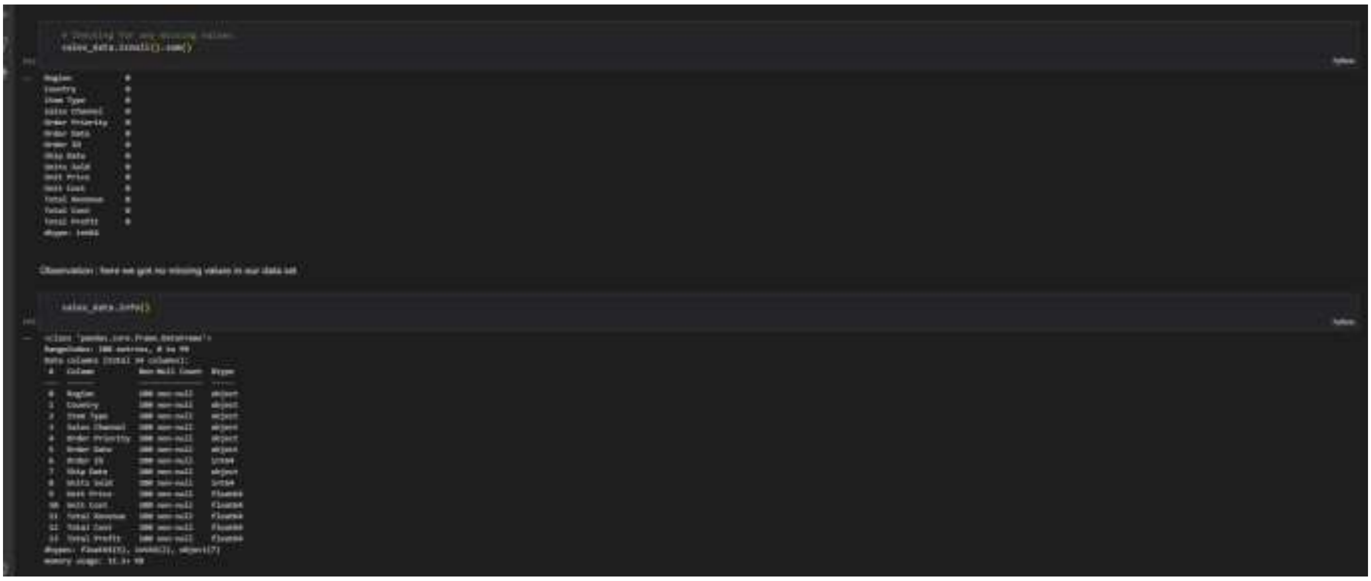
```
# Let's see the columns name
sales_data.columns
```

```
Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Priority',
       'Order Date', 'Order ID', 'Ship Date', 'Units Sold', 'Unit Price',
       'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit'],
      dtype='object')
```

```
sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Region          100 non-null    object 
 1   Country         100 non-null    object 
 2   Item Type       100 non-null    object 
 3   Sales Channel   100 non-null    object 
 4   Order Priority  100 non-null    object 
 5   Order Date      100 non-null    object 
 6   Order ID        100 non-null    int64  
 7   Ship Date       100 non-null    object 
 8   Units Sold      100 non-null    int64  
 9   Unit Price      100 non-null    float64
 10  Unit Cost       100 non-null    float64
 11  Total Revenue   100 non-null    float64
 12  Total Cost      100 non-null    float64
 13  Total Profit    100 non-null    float64
dtypes: float64(5), int64(2), object(7)
memory usage: 11.3+ KB
```

Observation: In the information of the dataset we got to know the number of columns, columns name and the data type of the columns.



```
# Checking for any missing values
sales_data.isnull().sum()
```

```
Region            0
Country           0
Item Type         0
Sales Channel     0
Order Priority    0
Order Date        0
Order ID          0
Ship Date         0
Units Sold        0
Unit Price        0
Unit Cost         0
Total Revenue     0
Total Cost        0
Total Profit      0
dtype: int64
```

Observation : here we got no missing values in our data set

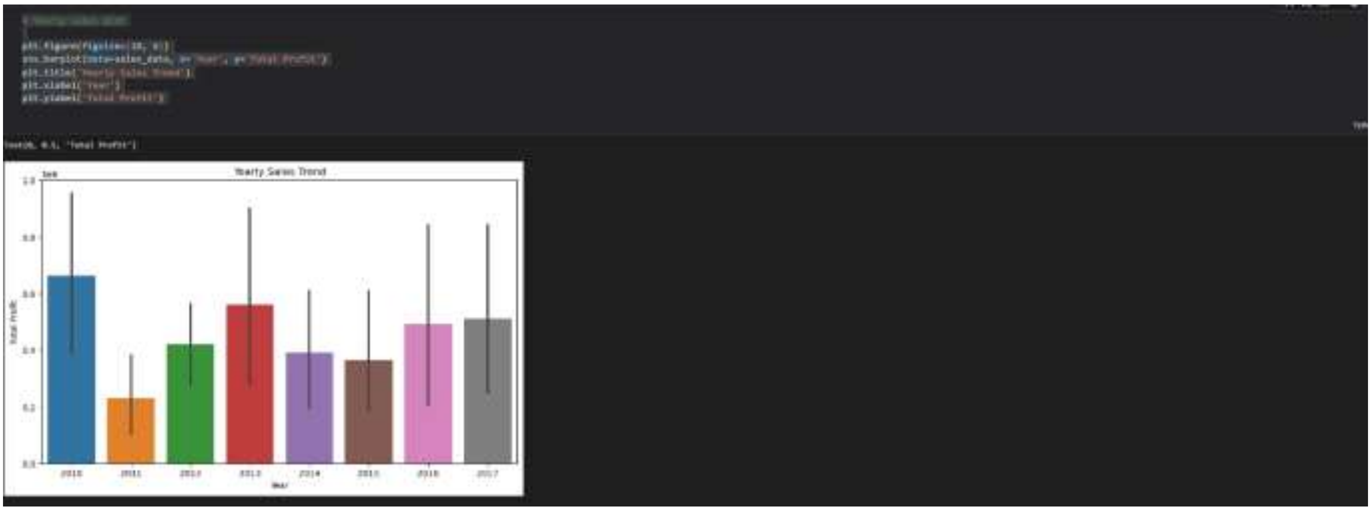
```
sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Region          100 non-null    object 
 1   Country         100 non-null    object 
 2   Item Type       100 non-null    object 
 3   Sales Channel   100 non-null    object 
 4   Order Priority  100 non-null    object 
 5   Order Date      100 non-null    object 
 6   Order ID        100 non-null    int64  
 7   Ship Date       100 non-null    object 
 8   Units Sold      100 non-null    int64  
 9   Unit Price      100 non-null    float64
 10  Unit Cost       100 non-null    float64
 11  Total Revenue   100 non-null    float64
 12  Total Cost      100 non-null    float64
 13  Total Profit    100 non-null    float64
dtypes: float64(5), int64(2), object(7)
memory usage: 11.3+ KB
```

Visualizing Sales Trend:-

Visualizing sales trends year and month wise can provide a more granular view of sales data and help identify patterns and seasonal trends. This type of analysis can be particularly useful for businesses that experience fluctuations in sales due to seasonal or other factors.

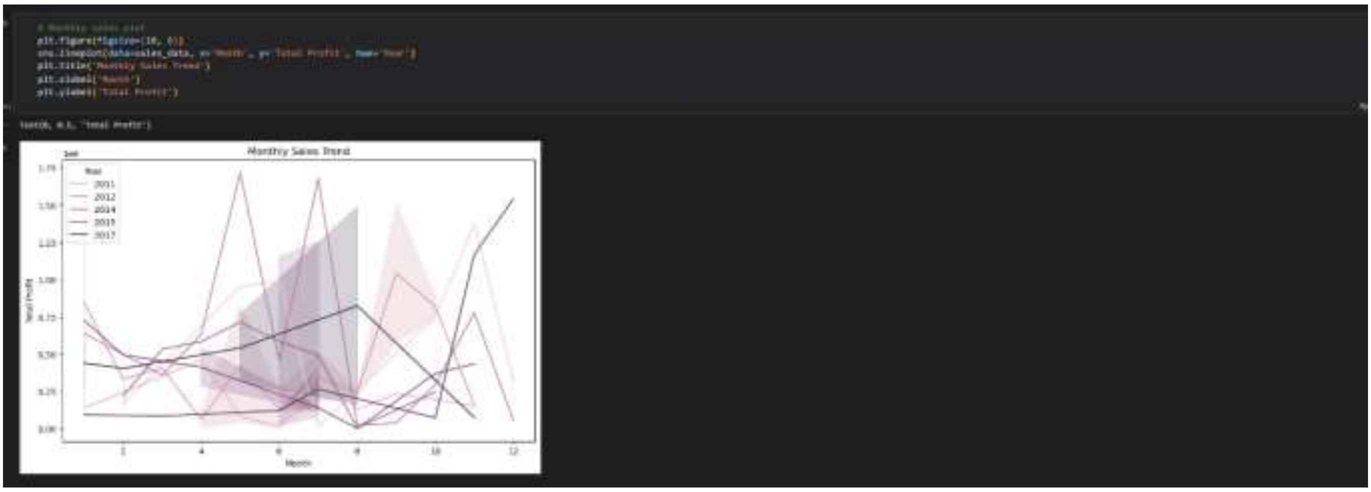
One way to visualize sales trends year and month wise is to use a line plot or a heatmap. A line plot can show how sales vary over time, while a heatmap can show which months are typically high or low in sales.



```
# Yearly sales plot

plt.figure(figsize=(10, 6))
sns.barplot(data=sales_data, x='Year', y='Total Profit')
plt.title('Yearly Sales Trend')
plt.xlabel('Year')
plt.ylabel('Total Profit')
```

```
Text(0, 0.5, 'Total Profit')
```



```
# Monthly sales plot
plt.figure(figsize=(10, 6))
sns.lineplot(data=sales_data, x='Month', y='Total Profit', hue='Year')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Profit')
```

```
Text(0, 0.5, 'Total Profit')
```