Academic Year: 2023-24 **Semester: VIII**
Class / Branch: BE Computer **Subject: Social Media Analytics Lab**

Name: Shivam Pandey
Date of performance: 24/02/2024
Date of Submission: 28/02/2024

Experiment No. 04

Aim: To perform exploratory data analysis and visualization of Social media data for business.

Objective: Uncover insights from social media data, including user behavior, sentiment, and trends, to inform strategic business decisions through visualization.

Software used: Kaggle Notebook.

Theory:
In today's competitive market landscape, understanding sales performance is essential for businesses to make informed decisions and strategize effectively. Exploratory Data Analysis (EDA) serves as a powerful tool for unraveling patterns, trends, and insights hidden within sales data. By analyzing various aspects such as sales distribution, revenue patterns, discount rates, geographic trends, deal sizes, and user engagement, companies can gain valuable insights to optimize their sales strategies and enhance overall performance.

Sales Distribution:
The distribution of sales across different product lines provides insights into the popularity and demand for various products offered by the company. By identifying top-performing product lines, businesses can allocate resources effectively, prioritize product development efforts, and tailor marketing strategies to capitalize on high-demand products.

Yearly and Monthly Revenues:
Analyzing yearly and monthly revenue trends enables businesses to identify seasonal patterns, peak sales periods, and areas of growth or decline. This information helps in budgeting, forecasting, and resource allocation to maximize revenue generation opportunities throughout the year.

Discount Rates and Sales Relationship:
Examining the relationship between discount rates and sales allows businesses to assess the effectiveness of discount strategies in driving sales volume and revenue. Understanding how discounts influence customer behavior and purchase decisions can inform pricing strategies and promotional campaigns to optimize sales performance while maintaining profitability.

Number of Sales for Countries and Territories:
Geographic analysis of sales data helps businesses identify regions with the highest sales volume, market potential, and growth opportunities. By understanding regional sales dynamics, companies can tailor marketing initiatives, expand into new markets, and optimize distribution channels to capitalize on geographic-specific trends and preferences.

Sales Distribution for Deal Sizes:
Analyzing the distribution of deal sizes provides insights into the typical transaction value and customer segmentation based on purchasing power. By segmenting customers according to deal sizes, businesses can customize sales strategies, offer personalized incentives, and enhance customer satisfaction to drive repeat purchases and long-term loyalty.

Monthly Active Users:
Tracking monthly active users enables businesses to gauge user engagement, retention rates, and overall customer satisfaction levels. By identifying patterns in user activity, businesses can optimize product offerings, user experience, and customer support services to foster continuous engagement and maximize customer lifetime value.

Implementation and Output:
In the below implementation, sales data is imported from Kaggle and a distance plot, histogram plot, line plot, bar graph, count plot, and box plot are executed to do Exploratory Data Analysis.

```
# Importing Packages

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import r2_score

# Input data files

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
# Loading the Data
sales = pd.read_csv('../input/sample-sales-data/sales_data_sample.csv',encoding='latin1')
sales.sort_values(by=['ORDERNUMBER'])
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	ADDI
578	10100	30	100.00	3	5151.00	1/6/2003 0:00	Shipped	1	1	2003	...	Airp
2024	10100	49	34.47	1	1689.03	1/6/2003 0:00	Shipped	1	1	2003	...	Airp
680	10100	50	67.80	2	3390.00	1/6/2003 0:00	Shipped	1	1	2003	...	Airp
1267	10100	22	86.51	4	1903.22	1/6/2003 0:00	Shipped	1	1	2003	...	Airp
728	10101	25	100.00	4	3782.00	1/9/2003 0:00	Shipped	1	1	2003	...	Ly
...	...	...	...	...	...	...	...	...	...	...	...	
2405	10425	18	100.00	2	1895.94	5/31/2005 0:00	In Process	2	5	2005	...	
393	10425	33	100.00	4	4692.60	5/31/2005 0:00	In Process	2	5	2005	...	
160	10425	38	100.00	12	5894.94	5/31/2005 0:00	In Process	2	5	2005	...	
780	10425	19	49.22	10	935.18	5/31/2005 0:00	In Process	2	5	2005	...	

```
# Rename columns in lowercase
sales.columns= sales.columns.str.lower()
```

```
sales.columns.values
```

```
array(['ordernumber', 'quantityordered', 'priceeach', 'orderlinenumber',
       'sales', 'orderdate', 'status', 'qtr_id', 'month_id', 'year_id',
       'productline', 'msrp', 'productcode', 'customername', 'phone',
       'addressline1', 'addressline2', 'city', 'state', 'postalcode',
       'country', 'territory', 'contactlastname', 'contactfirstname',
       'dealsize'], dtype=object)
```

```
sales.describe()
```

	ordernumber	quantityordered	priceeach	orderlinenumber	sales	qtr_id	month_id	year_id	msrp
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.00000	2823.000000
mean	10258.725115	35.092809	83.658544	6.466171	3553.889072	2.717676	7.092455	2003.81509	100.715551
std	92.085478	9.741443	20.174277	4.225841	1841.865106	1.203878	3.656633	0.69967	40.187912
min	10100.000000	6.000000	26.880000	1.000000	482.130000	1.000000	1.000000	2003.00000	33.000000
25%	10180.000000	27.000000	68.860000	3.000000	2203.430000	2.000000	4.000000	2003.00000	68.000000
50%	10262.000000	35.000000	95.700000	6.000000	3184.800000	3.000000	8.000000	2004.00000	99.000000
75%	10333.500000	43.000000	100.000000	9.000000	4508.000000	4.000000	11.000000	2004.00000	124.000000
max	10425.000000	97.000000	100.000000	18.000000	14082.800000	4.000000	12.000000	2005.00000	214.000000

```
# Checking null values
sales.isnull().sum()
```

```
ordernumber            0
quantityordered        0
priceeach              0
orderlinenumber        0
sales                  0
orderdate              0
status                 0
qtr_id                 0
month_id               0
year_id                0
productline            0
msrp                   0
productcode            0
customername           0
phone                  0
addressline1           0
addressline2        2521
city                   0
state               1486
postalcode            76
country                0
territory           1074
contactlastname        0
contactfirstname       0
dealsize               0
dtype: int64
```

```
# Checking duplicate values
len(sales) == len(sales.drop_duplicates())
```

```
True
```

```
# Determining Countries with null valued Territory
sales.loc[sales['territory'].isnull()]['country'].unique()
```

```
array(['USA', 'Canada'], dtype=object)
```

```
# Assigning North America Territory values
sales['territory'] = sales['territory'].fillna('NAM')
sales['territory'].unique()
```

```
array(['NAM', 'EMEA', 'APAC', 'Japan'], dtype=object)
```

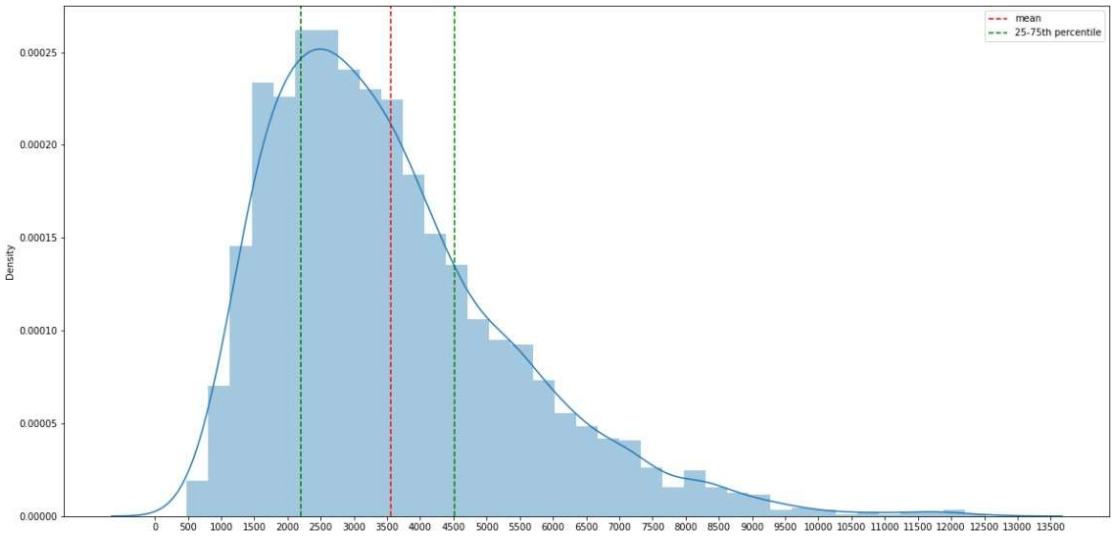
```
# Removing the In Process and Cancelled orders
sales['status'].unique()
```

```
array(['Shipped', 'Disputed', 'In Process', 'Cancelled', 'On Hold',
       'Resolved'], dtype=object)
```

```
sales1 = sales[~((sales['status'] == 'Cancelled') | (sales['status'] == 'On Hold'))]
sales1['status'].unique()
```

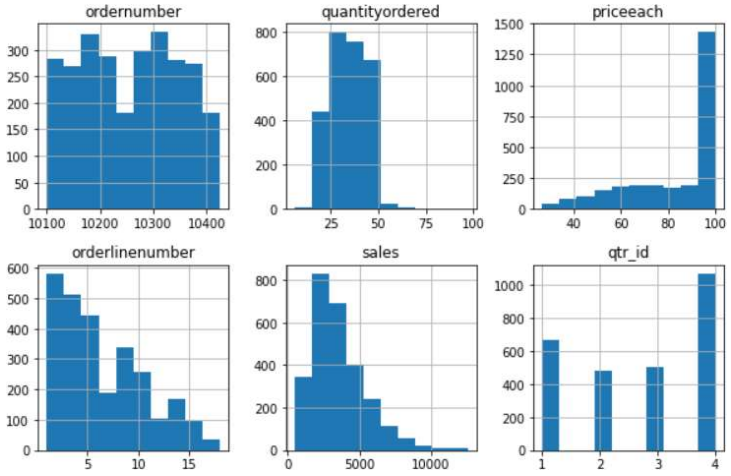
```
array(['Shipped', 'Disputed', 'In Process', 'Resolved'], dtype=object)
```

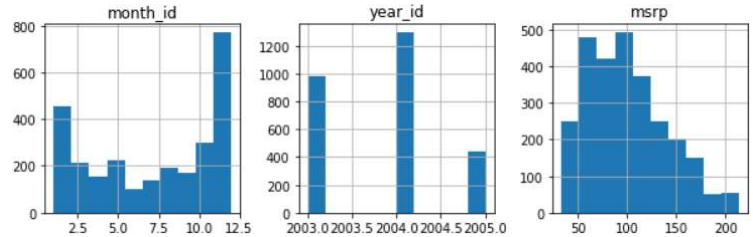
```
plt.figure(figsize=(20,10))
sns.distplot(x = sales1['sales'])
plt.axvline(x = np.mean(sales1['sales']), c = 'red', ls = '--', label = 'mean')
plt.axvline(x = np.percentile(sales1['sales'], 25), c = 'green', ls = '--', label = '25-75th percentile')
plt.axvline(x = np.percentile(sales1['sales'], 75), c = 'green', ls = '--')
plt.xticks(np.arange(0,14000,500))
plt.legend()
plt.show()
```



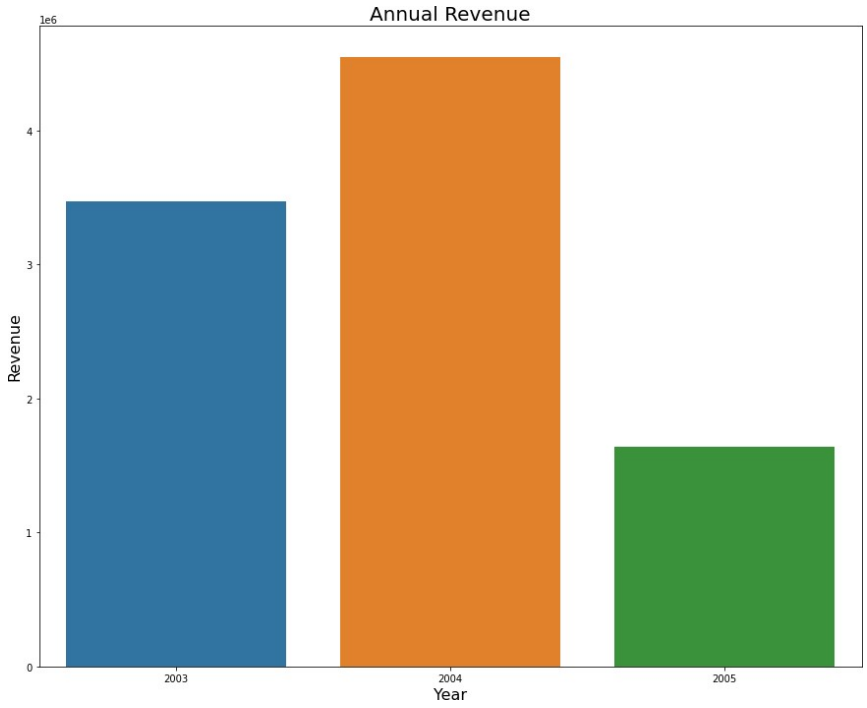
Sales distribution is right skewed. Most of the sales are between 2000-2500. Mean sale is around 3500.

```
sales1.hist(figsize = (10,10))
plt.show()
```



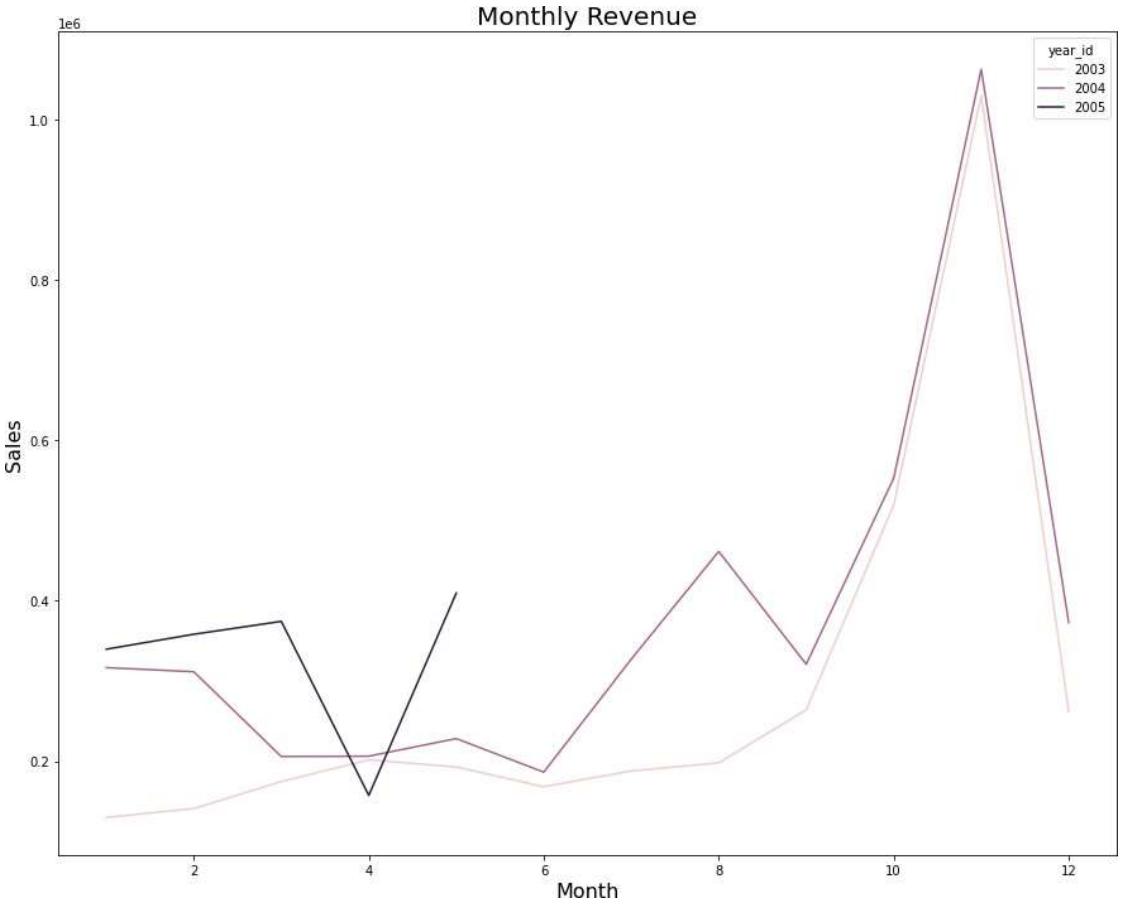



```
# Annual Revenue
plt.figure(figsize=(15,12))

#df1 = sales1.groupby('year_id').agg({'sales':'sum'}).rename(columns={'sales':'Revenue'})
#df1.reset_index(inplace=True)
#plt.bar(df1['year_id'],df1['Revenue'], width = 0.8)
yearly_revenue = sales1.groupby(['year_id'])['sales'].sum().reset_index()
sns.barplot(x="year_id", y="sales", data=yearly_revenue)

plt.title('Annual Revenue', fontsize = 20)
plt.xlabel('Year', fontsize = 16)
plt.ylabel('Revenue', fontsize = 16)

plt.show()
```



Sales are increased from 2003 to 2004. Sales at 2005 is least but it doesn't contain whole year data.

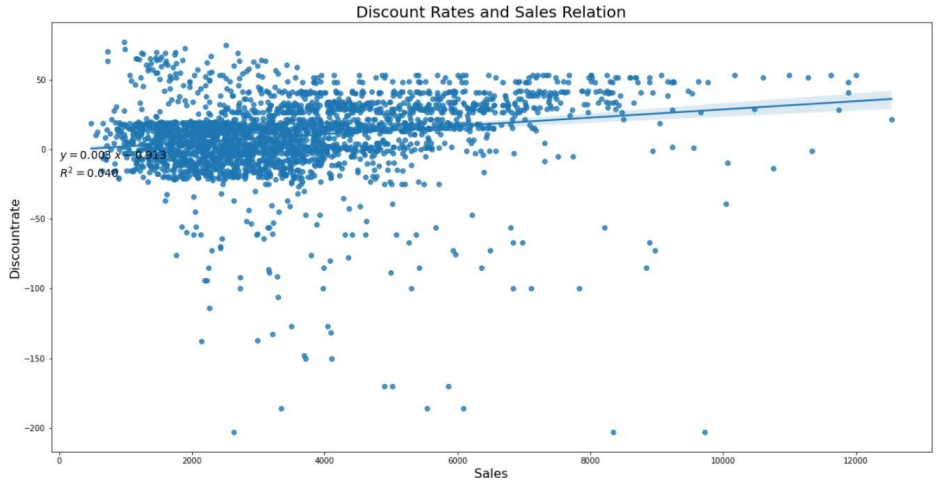
```
# Monthly Revenue
plt.figure(figsize=(15,12))

monthly_revenue = sales1.groupby(['year_id','month_id'])['sales'].sum().reset_index()

sns.lineplot(x="month_id", y="sales",hue="year_id", data=monthly_revenue)
plt.title('Monthly Revenue', fontsize = 20)
plt.xlabel('Month', fontsize = 16)
plt.ylabel('Sales', fontsize = 16)

plt.show()
```



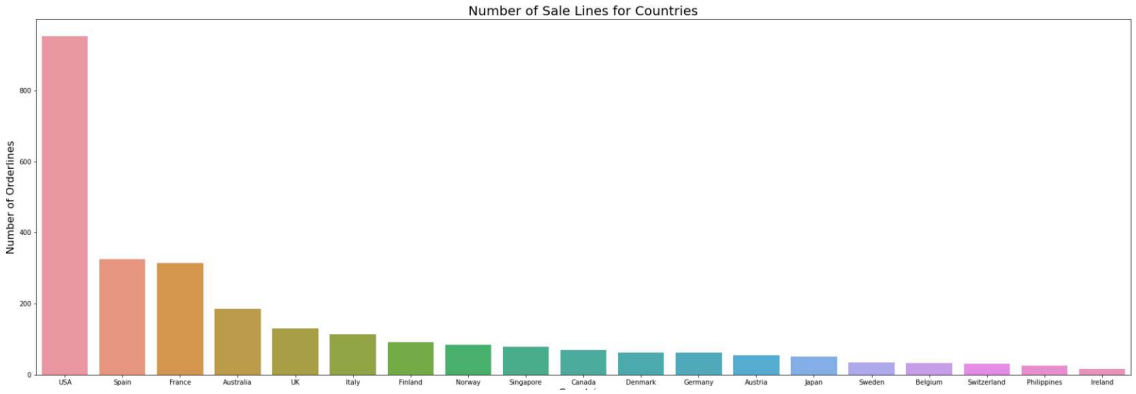

```
# Discount Rate
sales1['discountrate'] = ((sales1['msrp']-sales1['priceeach'])/sales1['msrp'])*100
sales1[['priceeach','msrp','discountrate']]
```

	priceeach	msrp	discountrate
0	95.70	95	-0.736842
1	81.35	95	14.368421
2	94.74	95	0.273684
3	83.26	95	12.357895
4	100.00	95	-5.263158
...	...	...	...
2817	97.16	54	-79.925926
2818	100.00	54	-85.185185
2819	100.00	54	-85.185185
2820	100.00	54	-85.185185
2821	62.24	54	-15.259259

```
# Discount Rates and Sales Relation
plt.figure(figsize=(20,10))
sns.regplot(x = sales1['sales'], y = sales1['discountrate'])
plt.title('Discount Rates and Sales Relation', fontsize = 20)
plt.ylabel('Discountrate', fontsize = 16)
plt.xlabel('Sales', fontsize = 16)


x = sales1['sales']
y = sales1['discountrate']
z = np.polyfit(x, y, 1)
p = np.poly1d(z)

text = f"$y={z[0]:0.3f}\;x{z[1]:+0.3f}$\n$R^2 = {r2_score(y,p(x)):0.3f}$"
plt.gca().text(0.05, 0.05, text, fontsize=14, verticalalignment='top')
plt.show()
```

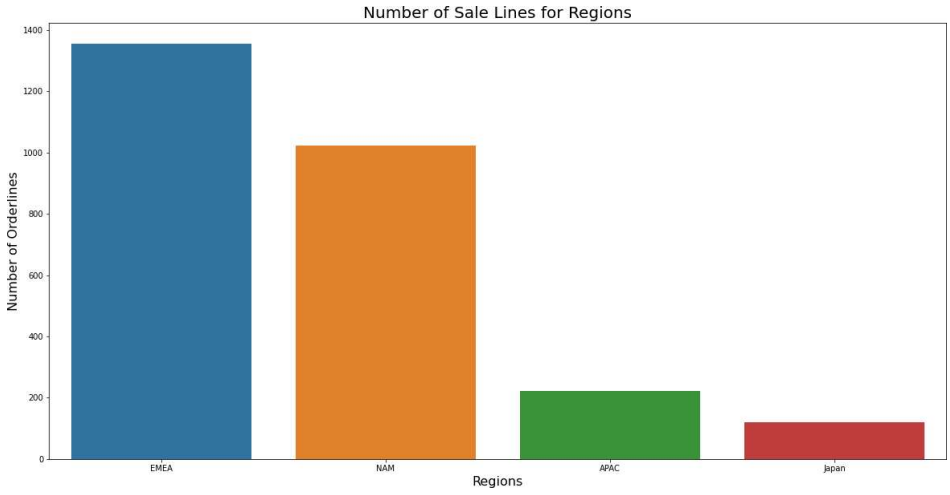



```
# Number of Sale Lines for Countries

plt.figure(figsize=(30,10))
sns.countplot(x = sales1['country'], order = sales1['country'].value_counts().index )
plt.title('Number of Sale Lines for Countries', fontsize = 20)
plt.ylabel('Number of Orderlines', fontsize = 16)
plt.xlabel('Countries', fontsize = 16)
plt.show()
```

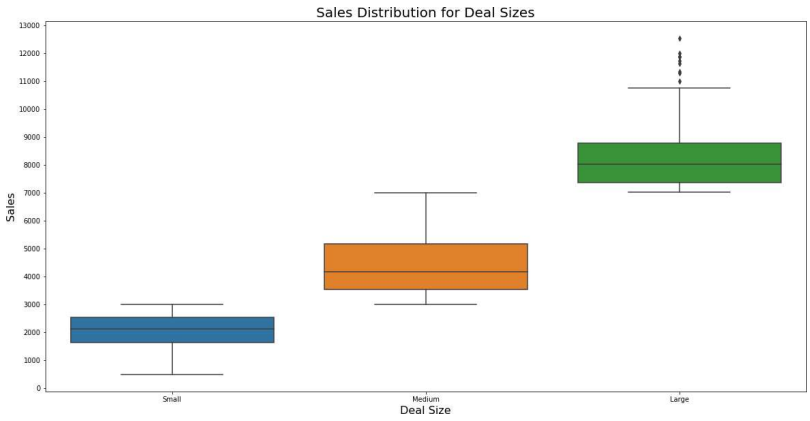


The most of sales are happened at USA. Spain and France are the second and third most sold countries

```
# Number of Sale Lines for Regions
plt.figure(figsize=(20,10))
sns.countplot(x = sales1['territory'], order = sales1['territory'].value_counts().index )
plt.title('Number of Sale Lines for Regions', fontsize = 20)
plt.ylabel('Number of Orderlines', fontsize = 16)
plt.xlabel('Regions', fontsize = 16)
plt.show()
```

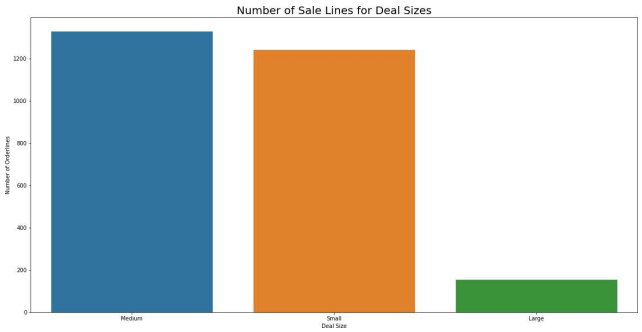


```
# Sales Distribution for Dealsizes
plt.figure(figsize=(20,10))
plt.yticks(np.arange(0,14000,1000))
sns.boxplot(x = sales1['dealsize'], y = sales1['sales'])
plt.title('Sales Distribution for Deal Sizes', fontsize = 20)
plt.ylabel('Sales', fontsize = 16)
plt.xlabel('Deal Size', fontsize = 16)
plt.show()
```



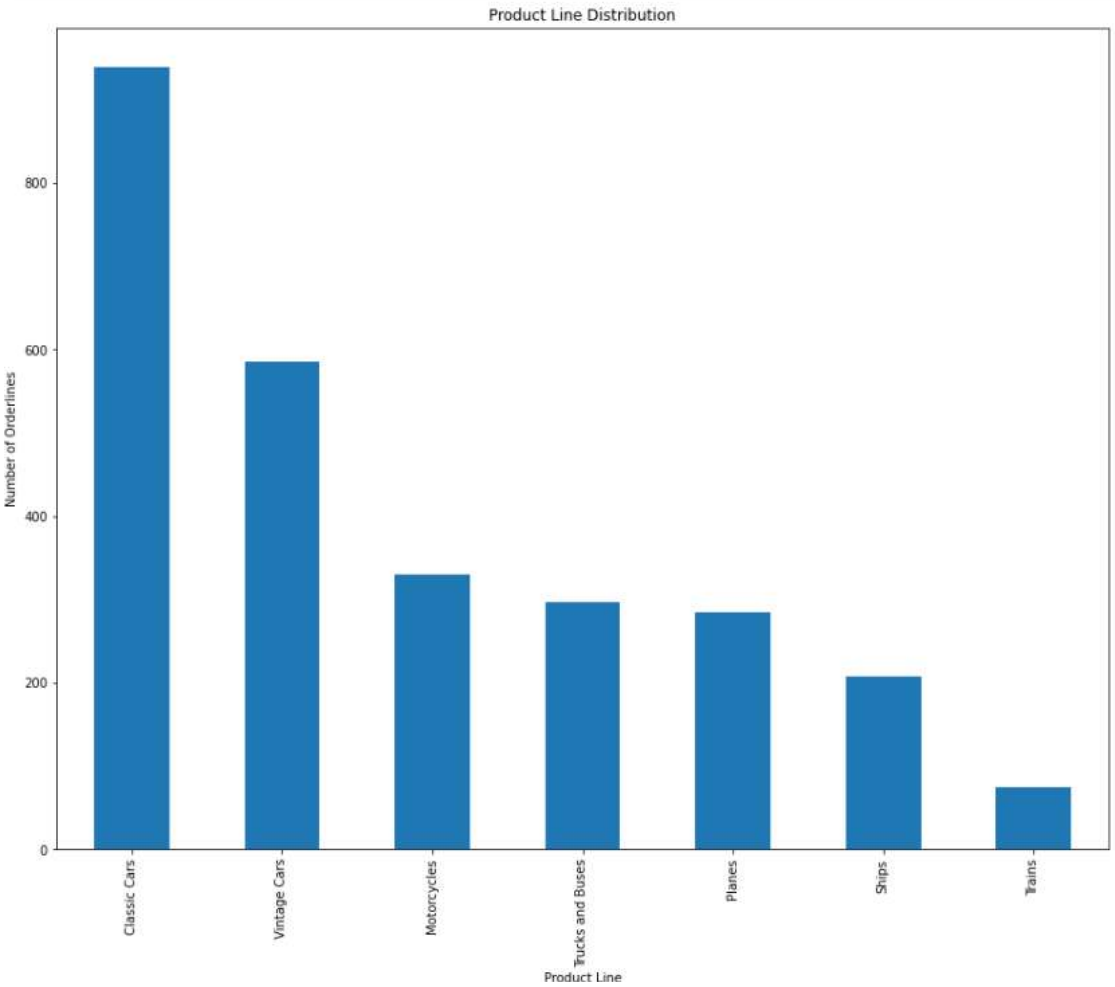
Small sales are sales between 0 and 3000. Medium sales are sales between 3000 and 7000. Large sales are the sales higher than 7000.

```
# Number of Sale Lines for Deal Size
plt.figure(figsize=(20,10))
sns.countplot(x = sales1['dealsize'], order = sales1['dealsize'].value_counts().index )
plt.title('Number of Sale Lines for Deal Sizes', fontsize = 20)
plt.ylabel('Number of Orderlines')
plt.xlabel('Deal Size')
plt.show()
```



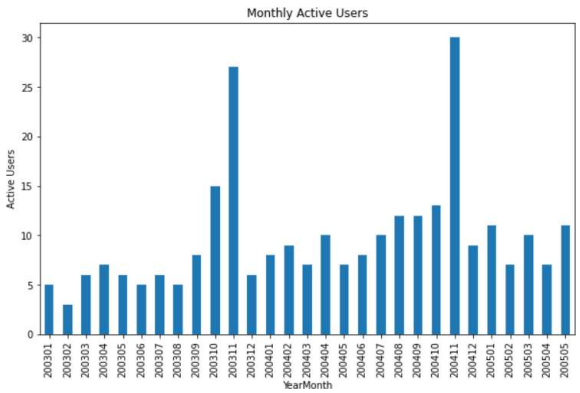
Small sales are sales between 0 and 3000. Medium sales are sales between 3000 and 7000. Large sales are the sales higher than 7000.

```
plt.figure(figsize=(15,12))
sales1['productline'].value_counts().plot(kind = 'bar')
plt.title('Product Line Distribution')
plt.xlabel('Product Line')
plt.ylabel('Number of Orderlines')
plt.show()
```



The most of the sales are belong Classic Cars category. Vintage Cars and Motorcycles are the second and third ones. The least of sales are happened in Trains category.

```
# Monthly Active Users
sales1['year_month'] = sales1['year_id'].map(str)+sales1['month_id'].map(str).map(lambda x: x.rjust(2,'0'))

plt.figure(figsize=(10,6))
sales1.groupby(['year_month'])['customername'].nunique().plot(kind='bar')
plt.title('Monthly Active Users')
plt.xlabel('YearMonth')
plt.ylabel('Active Users')
plt.show()
```



The monthly active users show similar trend to sales. The active users are peaked at most sold months.

Conclusion:

Exploratory Data Analysis (EDA) plays a pivotal role in unlocking the full potential of sales datasets. Through EDA, businesses gain a comprehensive understanding of their sales performance, uncovering nuanced patterns, correlations, and anomalies that might otherwise remain hidden. By delving into the intricacies of the dataset, EDA enables businesses to identify emerging trends, customer preferences, and market dynamics, providing valuable insights for strategic decision-making. Moreover, EDA empowers businesses to identify untapped opportunities and potential areas for improvement. Whether it's discovering underserved market segments, optimizing pricing strategies, or refining product offerings, EDA serves as a guiding compass for driving innovation and growth. Additionally, by proactively identifying and mitigating risks, businesses can navigate uncertainties with greater confidence, ensuring resilience and sustainability in a rapidly evolving marketplace.