Importing all the importanat libraries for Data Cleaning and EDA

In [101…

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

In [ ]:

In [102…

```
# Loading dataset into notebook
```

In [103…

```
df = pd.read_csv('sales_data_sample.csv',encoding='ISO-8859-1')
```

In [ ]:

In [104…

```
# Checking first five of dataset
```

In [105…

```
df.head()
```

Out[105…

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDEF
0	10107	30	95.70	2	2871.00	2/24
1	10121	34	81.35	5	2765.90	5/7
2	10134	41	94.74	2	3884.34	7/1
3	10145	45	83.26	6	3746.70	8/25
4	10159	49	100.00	14	5205.27	10/10

5 rows × 25 columns

In [ ]:

In [106…

```
# last five rows of dataset
```

In [107…

```
df.tail()
```

Out[107…

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	OR
2818	10350	20	100.00	15	2244.40	
2819	10373	29	100.00	1	3978.51	
2820	10386	43	100.00	4	5417.57	
2821	10397	34	62.24	1	2116.16	
2822	10414	47	65.52	9	3079.44	

5 rows × 25 columns

In [ ]:

In [108…
```
# Total Number of rows and cloumns presents in dataset
```

In [109…
```
df.shape
```

Out[109…
```
(2823, 25)
```

In [ ]:

In [110…
```
# checking datatype of columns
```

In [111…
```
df.dtypes
```

Out[111…

```
ORDERNUMBER           int64
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
ORDERDATE            object
STATUS               object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
PRODUCTLINE          object
MSRP                  int64
PRODUCTCODE          object
CUSTOMERNAME         object
PHONE                object
ADDRESSLINE1         object
ADDRESSLINE2         object
CITY                 object
STATE                object
POSTALCODE           object
COUNTRY              object
TERRITORY            object
CONTACTLASTNAME      object
CONTACTFIRSTNAME     object
DEALSIZE             object
dtype: object
```

In [ ]:

```
```

In [112…

```
# An Overview on dataset
```

In [113…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

Performing Data Cleaning

Droping NAN values

In [114…
```
# Total number of null values in particular columns
```

In [115…
```
df.isnull().sum()
```

Out[115…

```
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
```

In [ ]:

In [116…

```
# checking total % of null values presnt in overall dataset
```

In [117…

```
(df.isnull().sum().sum())/(df.shape[0]*df.shape[1])*100
```

Out[117…

```
7.30712008501594
```

In [118…

```
# Total 7% of data is missing from the dataset
```

In [ ]:

In [119…

```
# Checking % of null value in each column
```

In [120…

```
(df.isnull().sum()/df.shape[0])*100
```

Out[120…
```
ORDERNUMBER          0.000000
QUANTITYORDERED      0.000000
PRICEEACH            0.000000
ORDERLINENUMBER      0.000000
SALES                0.000000
ORDERDATE            0.000000
STATUS               0.000000
QTR_ID               0.000000
MONTH_ID             0.000000
YEAR_ID              0.000000
PRODUCTLINE          0.000000
MSRP                 0.000000
PRODUCTCODE          0.000000
CUSTOMERNAME         0.000000
PHONE                0.000000
ADDRESSLINE1         0.000000
ADDRESSLINE2        89.302161
CITY                 0.000000
STATE               52.639036
POSTALCODE           2.692171
COUNTRY              0.000000
TERRITORY           38.044633
CONTACTLASTNAME      0.000000
CONTACTFIRSTNAME     0.000000
DEALSIZE             0.000000
dtype: float64
```

In [121…
```
# Here we got to know that in
# column          % of null value
# ADDRESSLINE2       89.302161
# STATE              52.639036
# POSTALCODE         2.692171
# TERRITORY          38.044633
# % of null values presents
```

In [ ]:
```

```

In [122…
```
# if our column contain more than 80-90 percentage of null values then we have to d
# b'coz fill this much of data manually can give inaccurate output
```

In [123…
```
# Droping column
df.drop('ADDRESSLINE2',axis = 1,inplace=True)
```

In [124…
```
df.columns
```

Out[124…
```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
       'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
       'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
       'ADDRESSLINE1', 'CITY', 'STATE', 'POSTALCODE', 'COUNTRY', 'TERRITORY',
       'CONTACTLASTNAME', 'CONTACTFIRSTNAME', 'DEALSIZE'],
      dtype='object')
```

In [125…
```
# Sucessfully drop the column
```

```
In [ ]:
```

Filling of NAN values

```
In [126…  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  CITY              2823 non-null   object 
 17  STATE             1337 non-null   object 
 18  POSTALCODE        2747 non-null   object 
 19  COUNTRY           2823 non-null   object 
 20  TERRITORY         1749 non-null   object 
 21  CONTACTLASTNAME   2823 non-null   object 
 22  CONTACTFIRSTNAME  2823 non-null   object 
 23  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(15)
memory usage: 529.4+ KB
```

```
In [127…  df.isnull().sum()
```

Out[127…

```
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
CITY                    0
STATE                1486
POSTALCODE             76
COUNTRY                 0
TERRITORY            1074
CONTACTLASTNAME         0
CONTACTFIRSTNAME        0
DEALSIZE                0
dtype: int64
```

In [128…

```
# Here the columns which contain null values are of object datatypes
```

In [129…

```
for i  in df.select_dtypes(include= 'object').columns:
    df[i].fillna(df[i].mode()[0],inplace=True)
```

```
C:\Users\ankit\AppData\Local\Temp\ipykernel_16820\2214743516.py:2: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[i].fillna(df[i].mode()[0],inplace=True)
```

In [130…

```
df.isnull().sum()
```

Out[130…
```
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
CITY                0
STATE               0
POSTALCODE          0
COUNTRY             0
TERRITORY           0
CONTACTLASTNAME     0
CONTACTFIRSTNAME    0
DEALSIZE            0
dtype: int64
```

In [131…
```
# We sucessfully able to fill missing values
```

In [ ]:

Check for Duplicates

In [132…
```
df.duplicated().sum()
```

Out[132…
```
0
```

In [ ]:

Convert Data Types: Ensure columns like dates and numerical values are in the correct format.

In [133…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  CITY              2823 non-null   object 
 17  STATE             2823 non-null   object 
 18  POSTALCODE        2823 non-null   object 
 19  COUNTRY           2823 non-null   object 
 20  TERRITORY         2823 non-null   object 
 21  CONTACTLASTNAME   2823 non-null   object 
 22  CONTACTFIRSTNAME  2823 non-null   object 
 23  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(15)
memory usage: 529.4+ KB
```

In [134…

```
# ORDERDATE column should be in date format but it is in object
```

In [135…

```
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
```

In [157…

```
# Drive year from ORDERDATE column
df['YEAR'] = df['ORDERDATE'].dt.year
df['YEAR'] = df['YEAR'].round().astype(int)
```

In [137…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   ORDERNUMBER       2823 non-null   int64         
 1   QUANTITYORDERED   2823 non-null   int64         
 2   PRICEEACH         2823 non-null   float64       
 3   ORDERLINENUMBER   2823 non-null   int64         
 4   SALES             2823 non-null   float64       
 5   ORDERDATE         2823 non-null   datetime64[ns]
 6   STATUS            2823 non-null   object        
 7   QTR_ID            2823 non-null   int64         
 8   MONTH_ID          2823 non-null   int64         
 9   YEAR_ID           2823 non-null   int64         
 10  PRODUCTLINE       2823 non-null   object        
 11  MSRP              2823 non-null   int64         
 12  PRODUCTCODE       2823 non-null   object        
 13  CUSTOMERNAME      2823 non-null   object        
 14  PHONE             2823 non-null   object        
 15  ADDRESSLINE1      2823 non-null   object        
 16  CITY              2823 non-null   object        
 17  STATE             2823 non-null   object        
 18  POSTALCODE        2823 non-null   object        
 19  COUNTRY           2823 non-null   object        
 20  TERRITORY         2823 non-null   object        
 21  CONTACTLASTNAME   2823 non-null   object        
 22  CONTACTFIRSTNAME  2823 non-null   object        
 23  DEALSIZE          2823 non-null   object        
 24  YEAR              2823 non-null   int32         
dtypes: datetime64[ns](1), float64(2), int32(1), int64(7), object(14)
memory usage: 540.5+ KB
```

In [ ]:

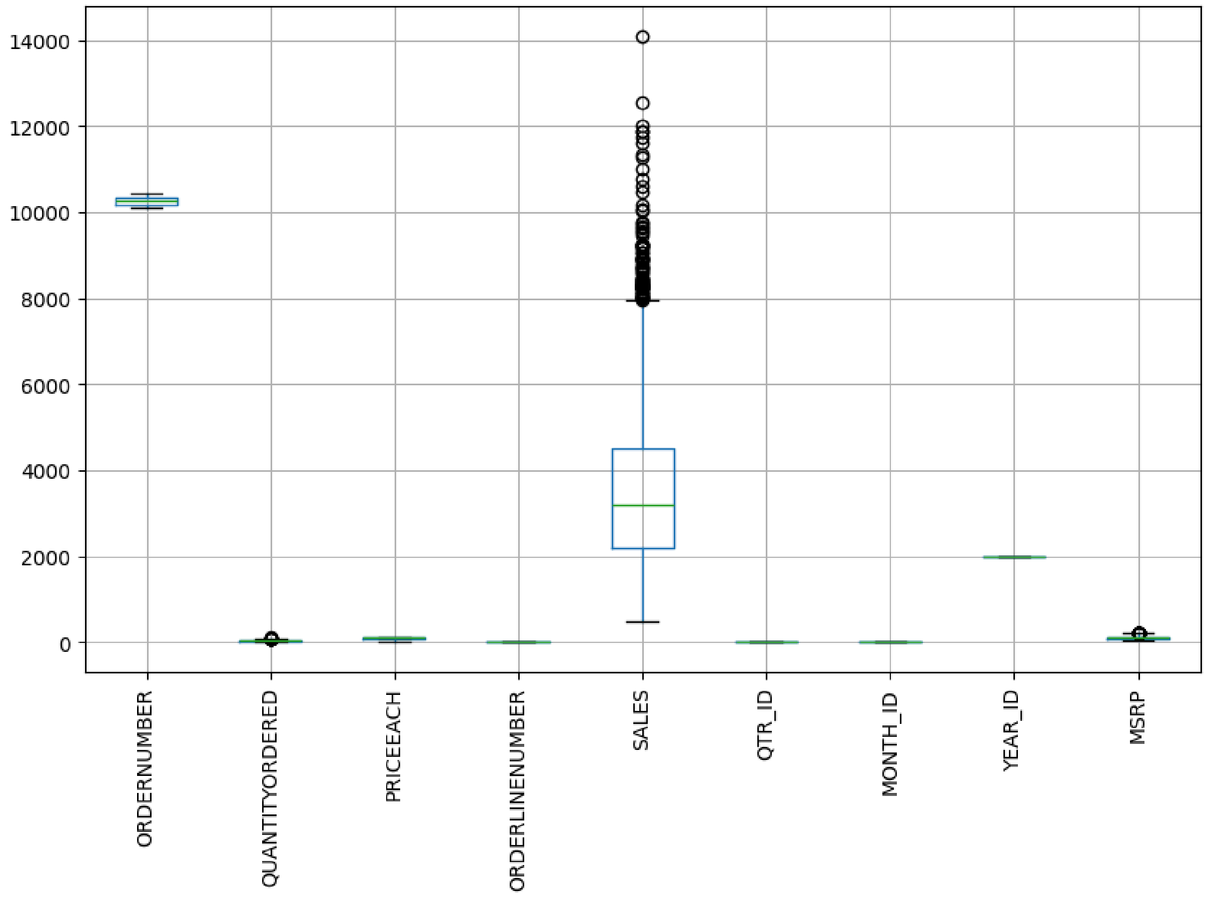
Handle Outliers:

In [138…

```
# For multiple columns
```

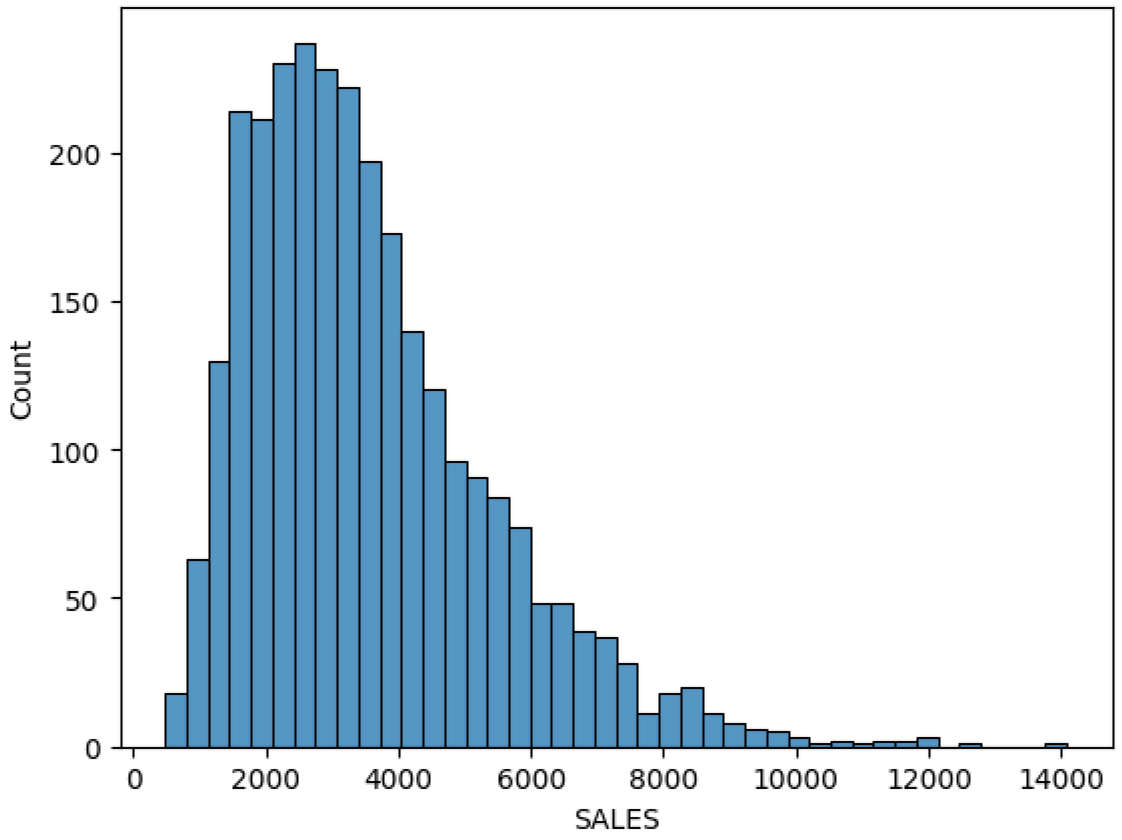
In [139…

```
df.select_dtypes(include=['int64', 'float64']).boxplot(figsize=(10, 6))
plt.xticks(rotation=90)
plt.show()
```



In [140…
```
# For single columns
```

In [141…
```
sns.histplot(df['SALES'])
plt.show()
```



```
In [ ]:
```

Performing Summary of Statistics

```
In [142…  # For continues datatype
```

```
In [143…  df.describe()
```

Out[143…

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SAL
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.0000
mean	10258.725115	35.092809	83.658544	6.466171	3553.8890
min	10100.000000	6.000000	26.880000	1.000000	482.1300
25%	10180.000000	27.000000	68.860000	3.000000	2203.4300
50%	10262.000000	35.000000	95.700000	6.000000	3184.8000
75%	10333.500000	43.000000	100.000000	9.000000	4508.0000
max	10425.000000	97.000000	100.000000	18.000000	14082.8000
std	92.085478	9.741443	20.174277	4.225841	1841.8651

In [144…

```
# for categorical datatype
```

In [145…

```
df.describe(include='object')
```

Out[145…

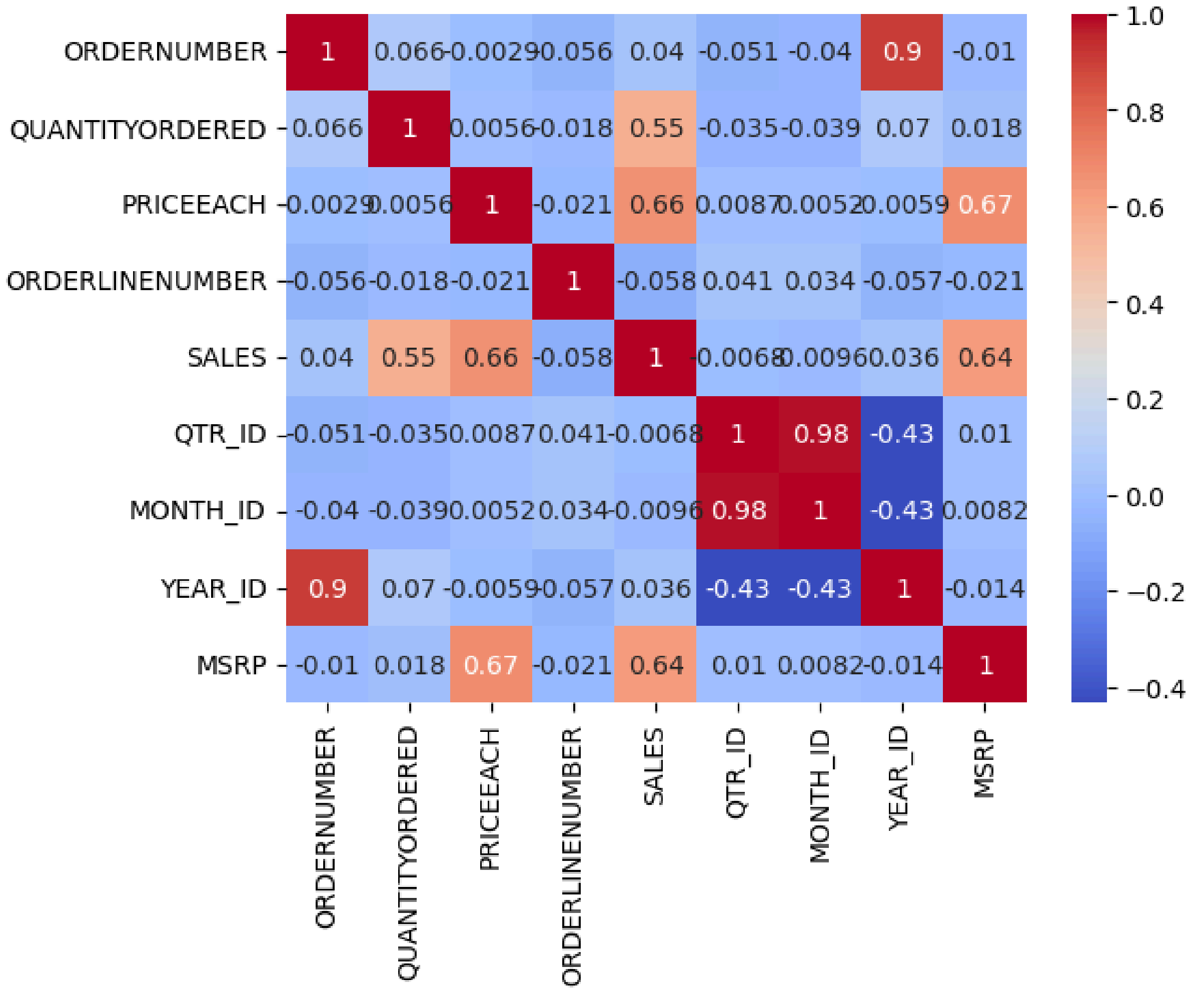
	STATUS	PRODUCTLINE	PRODUCTCODE	CUSTOMERNAME	PHONE	ADDRESSLINE
count	2823	2823	2823	2823	2823	282
unique	6	7	109	92	91	9
top	Shipped	Classic Cars	S18_3232	Euro Shopping Channel	(91) 555 94 44	C/ Moralzarza 8
freq	2617	967	52	259	259	25

In [146…

```
# Correlations: Analyzing relationships between sales and other numerical variables
```

In [147…

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df.select_dtypes(include=['int64', 'float64']).corr(), annot=True, cmap
plt.show()
```



	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	QTR_ID	MONTH_ID	YEAR_ID	MSRP
ORDERNUMBER	1	0.066	-0.0029	-0.056	0.04	-0.051	-0.04	0.9	-0.01
QUANTITYORDERED	0.066	1	0.0056	-0.018	0.55	-0.035	-0.039	0.07	0.018
PRICEEACH	-0.0029	0.0056	1	-0.021	0.66	0.0087	0.0052	-0.0059	0.67
ORDERLINENUMBER	-0.056	-0.018	-0.021	1	-0.058	0.041	0.034	-0.057	-0.021
SALES	0.04	0.55	0.66	-0.058	1	-0.0068	-0.0096	0.036	0.64
QTR_ID	-0.051	-0.035	0.0087	0.041	-0.0068	1	0.98	-0.43	0.01
MONTH_ID	-0.04	-0.039	0.0052	0.034	-0.0096	0.98	1	-0.43	0.0082
YEAR_ID	0.9	0.07	-0.0059	-0.057	0.036	-0.43	-0.43	1	-0.014
MSRP	-0.01	0.018	0.67	-0.021	0.64	0.01	0.0082	-0.014	1

In [ ]:

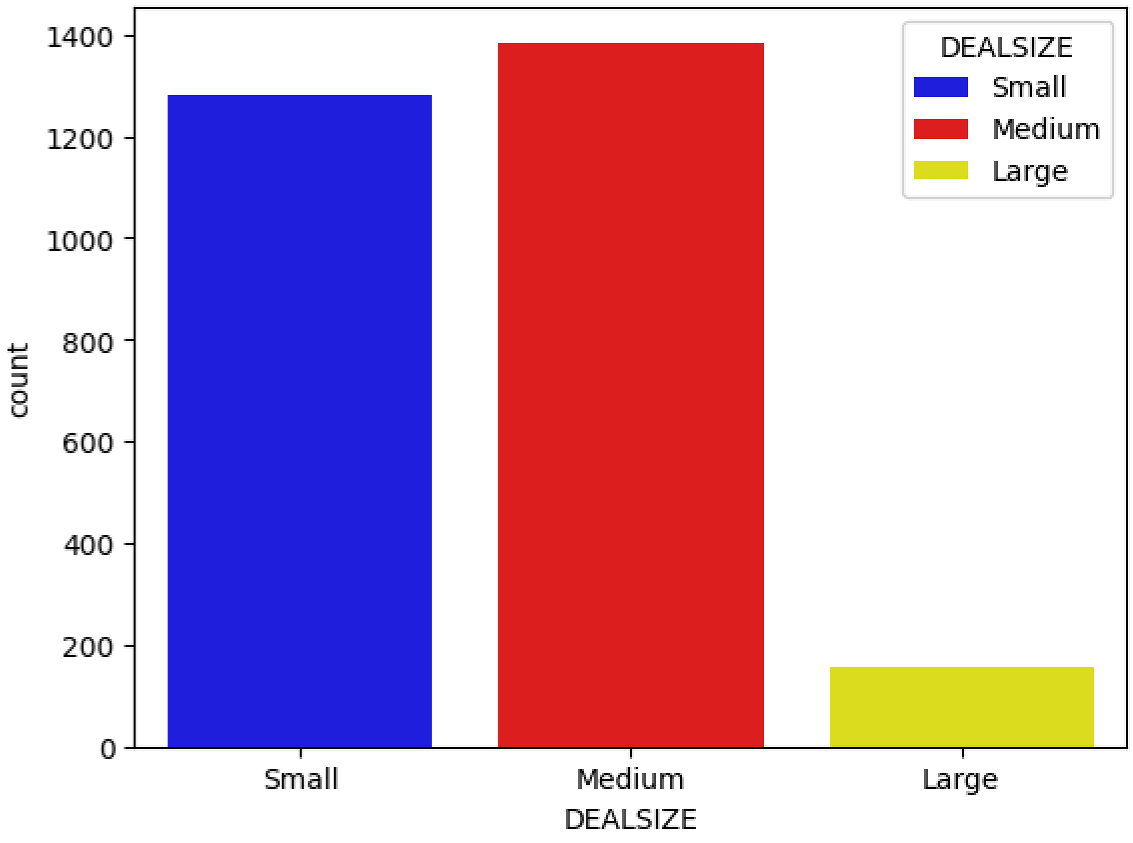
Performing Exploratory Data Analysis

Univariate Analysis

In [148…

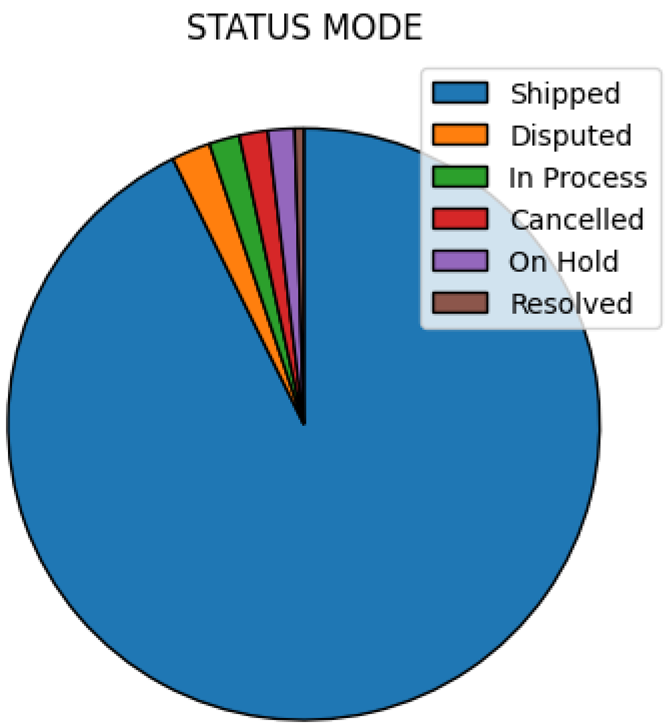
```
sns.countplot(x = df['DEALSIZE'],palette=['blue', 'red', 'yellow'])
plt.legend(title="DEALSIZE", labels=df['DEALSIZE'].unique())
plt.show()
```

```
C:\Users\ankit\AppData\Local\Temp\ipykernel_16820\4037968563.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x = df['DEALSIZE'],palette=['blue', 'red', 'yellow'])
```



In [ ]:
```
# Medium dealsize having high revenue
```

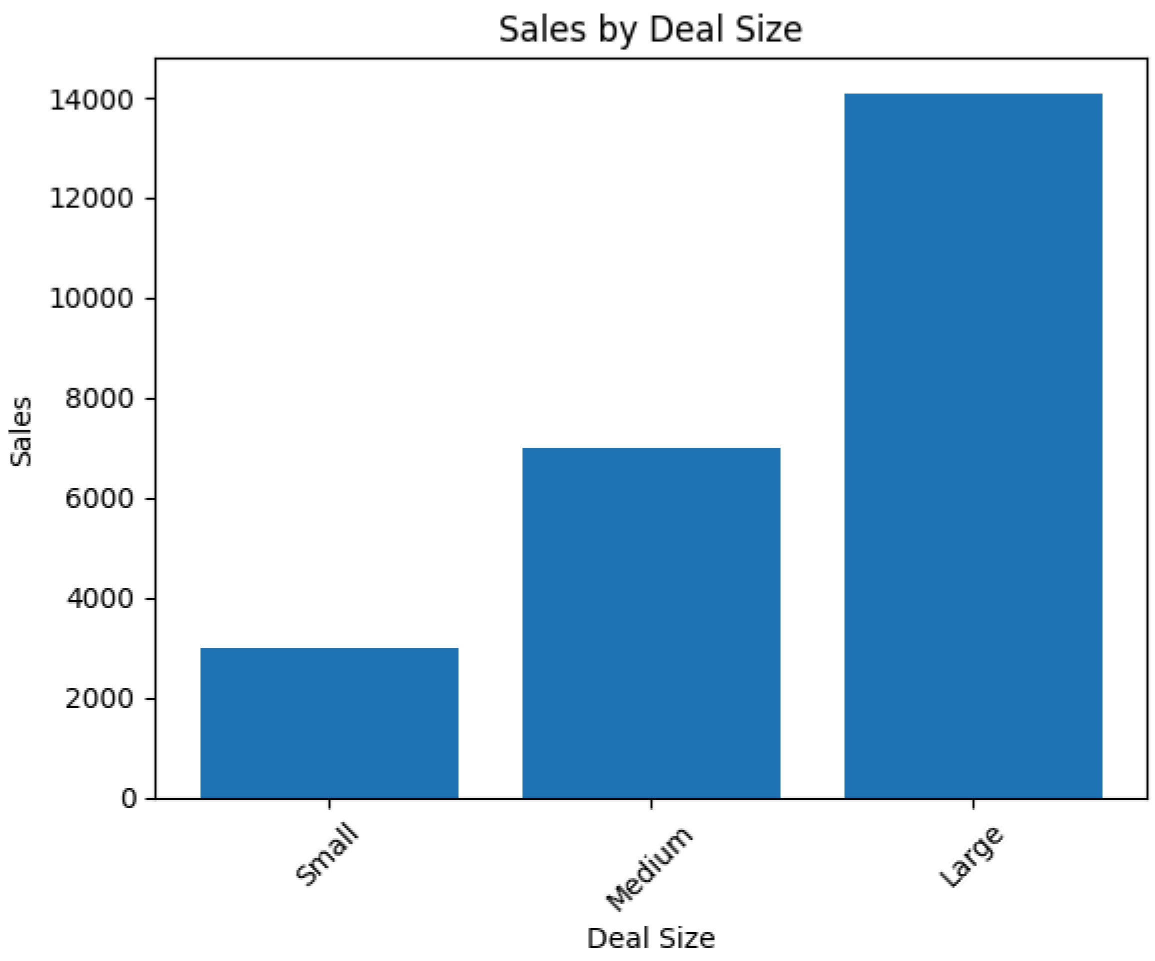
In [149…
```
plt.pie(df['STATUS'].value_counts(),startangle=90,counterclock=False,wedgeprops={'e
plt.title("STATUS MODE")
plt.legend(df['STATUS'].unique())
plt.show()
```



In [198…
```
# Shipped mode have best performance
```

Bivariate Analysis

In [150…
```
plt.bar(df['DEALSIZE'], df['SALES'])
plt.xlabel('Deal Size')
plt.ylabel('Sales')
plt.title('Sales by Deal Size')
plt.xticks(rotation=45)
plt.show()
```

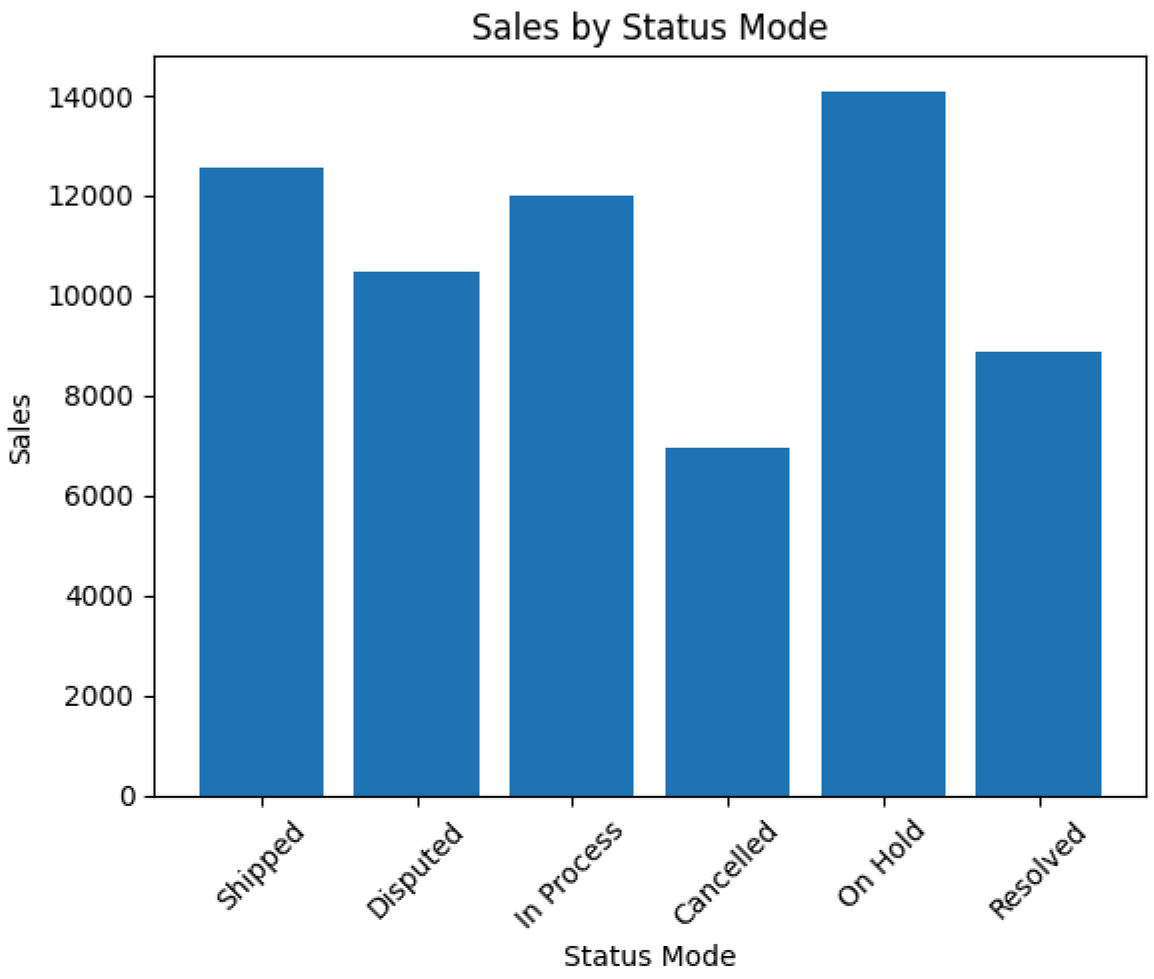


In [199…

```
# large deal sixe have high sales
```

In [151…

```
plt.bar(df['STATUS'], df['SALES'])
plt.xlabel('Status Mode')
plt.ylabel('Sales')
plt.title('Sales by Status Mode')
plt.xticks(rotation=45)
plt.show()
```

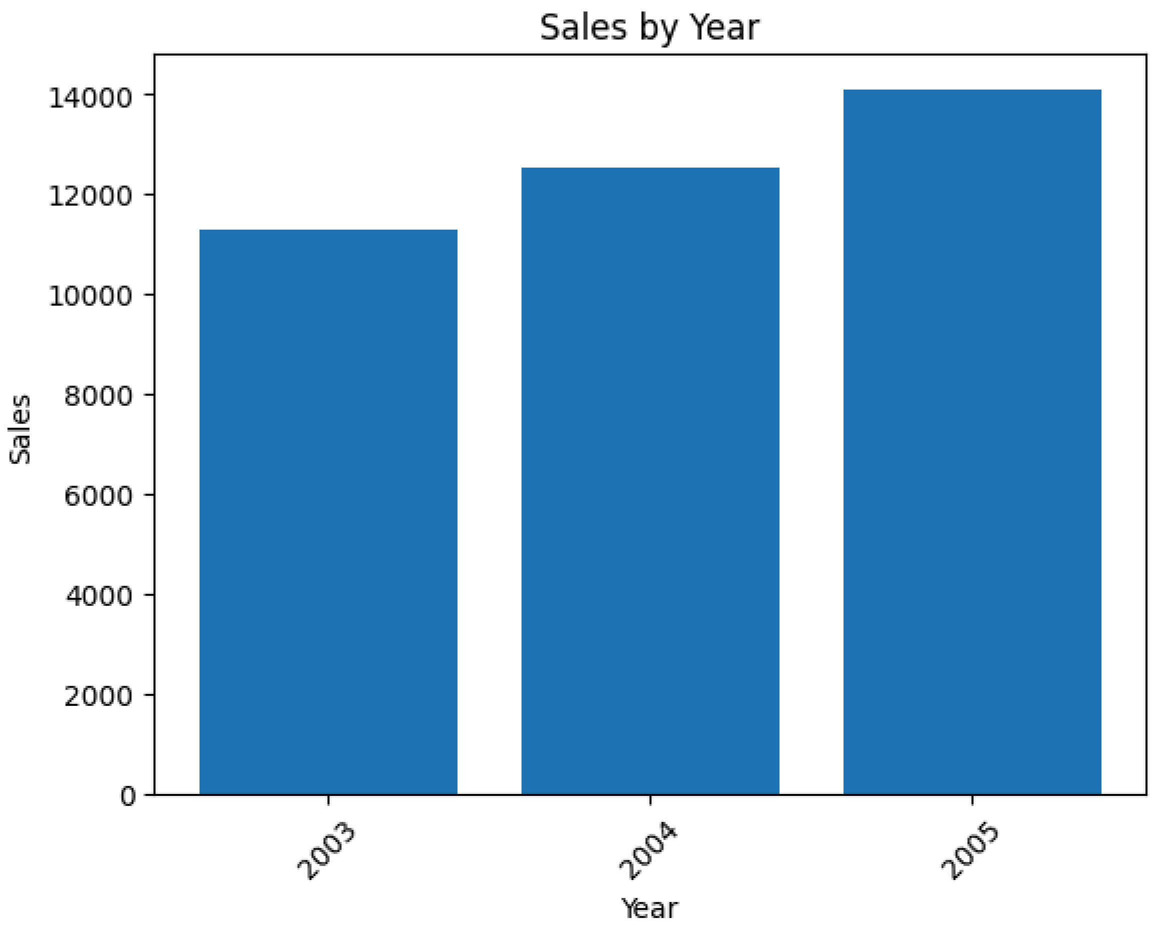


In [161…
```
df['YEAR'].replace({2004:'2004',2003:'2003',2005:'2005'},inplace=True)
```

```
C:\Users\ankit\AppData\Local\Temp\ipykernel_16820\3890760627.py:1: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['YEAR'].replace({2004:'2004',2003:'2003',2005:'2005'},inplace=True)
```

In [167…
```
# I replace year with string because my year value come as float like
# 2003.5  2005.0 2004.05
# for better visulization i do this
```

In [162…
```
plt.bar(df['YEAR'], df['SALES'])
plt.xlabel('Year')
plt.ylabel('Sales')
plt.title('Sales by Year')
plt.xticks(rotation=45)
plt.show()
```

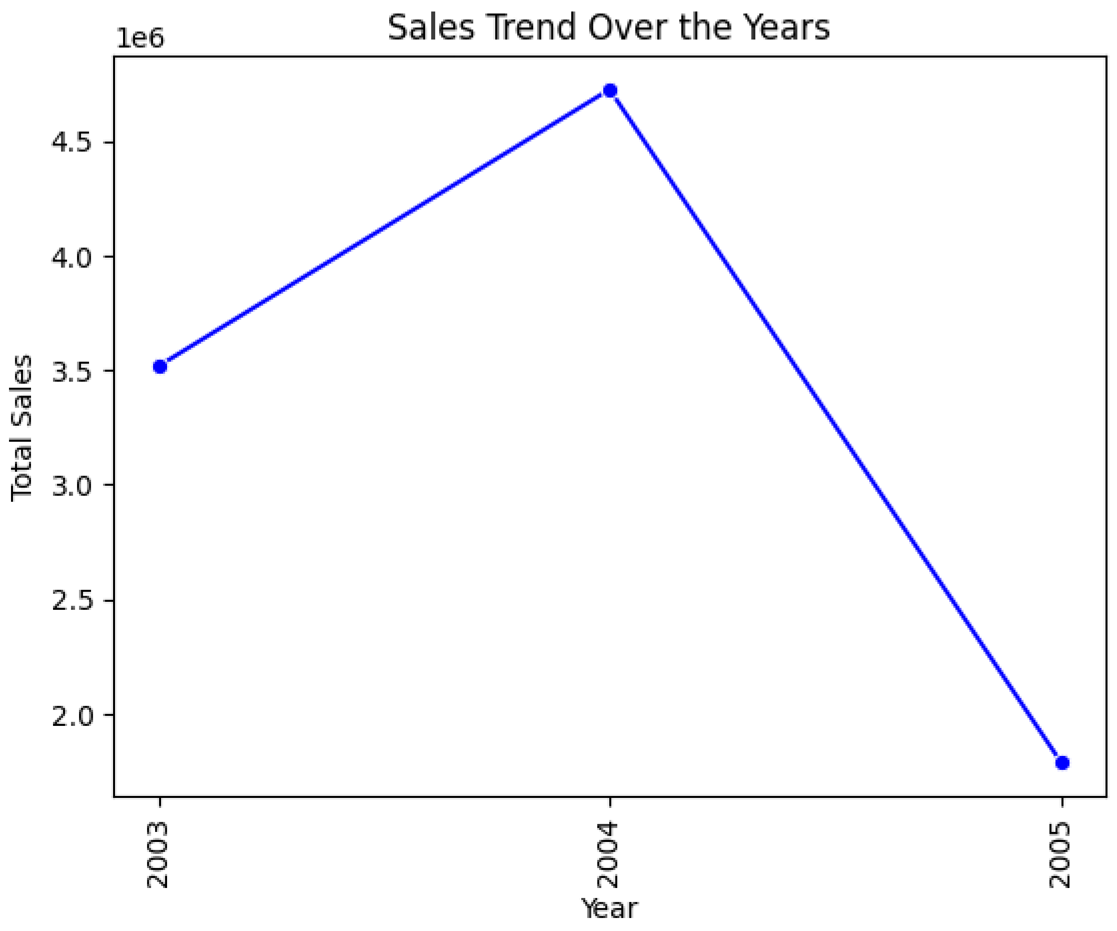


In [200…

```
# in 2005 year company generate high revenue
```

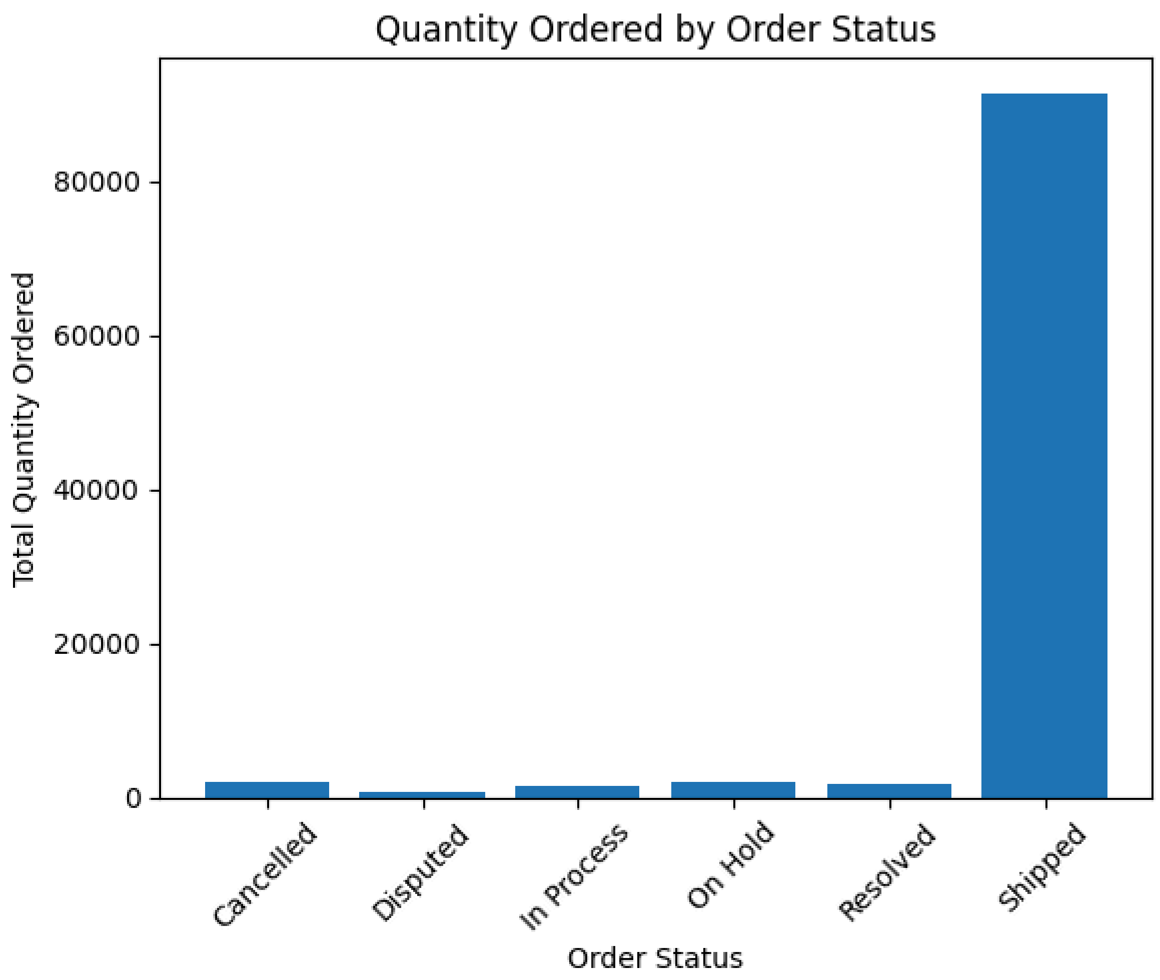
In [163…

```
yearly_sales = df.groupby('YEAR')['SALES'].sum().reset_index()
sns.lineplot(x=yearly_sales['YEAR'], y=yearly_sales['SALES'], marker='o', color='bl
plt.xlabel('Year')
plt.xticks(rotation=90)
plt.ylabel('Total Sales')
plt.title('Sales Trend Over the Years')
plt.show()
```



In [171…

```
status_quantity = df.groupby('STATUS')['QUANTITYORDERED'].sum()
plt.bar(status_quantity.index, status_quantity.values)

plt.xlabel('Order Status')
plt.ylabel('Total Quantity Ordered')
plt.title('Quantity Ordered by Order Status')
plt.xticks(rotation=45)
plt.show()
```

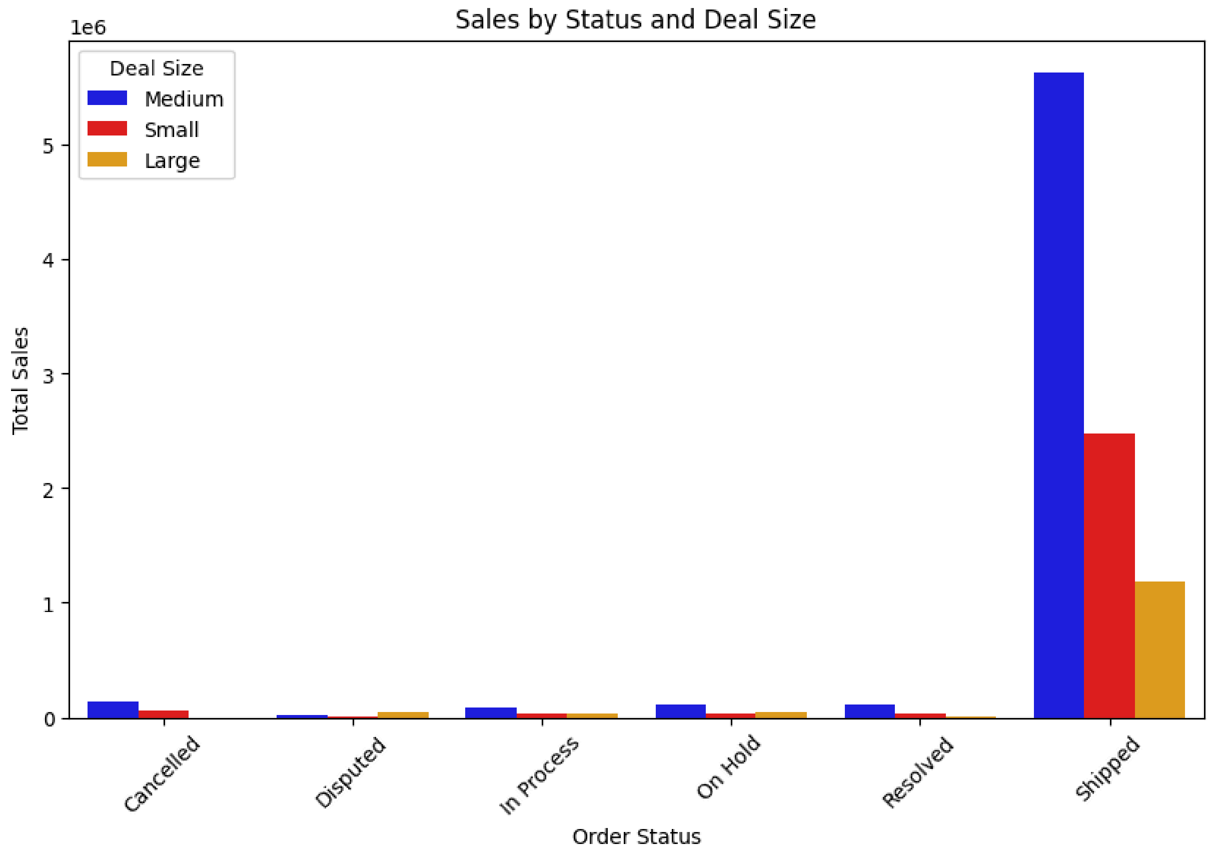


In [ ]:

Multivariate Analysis

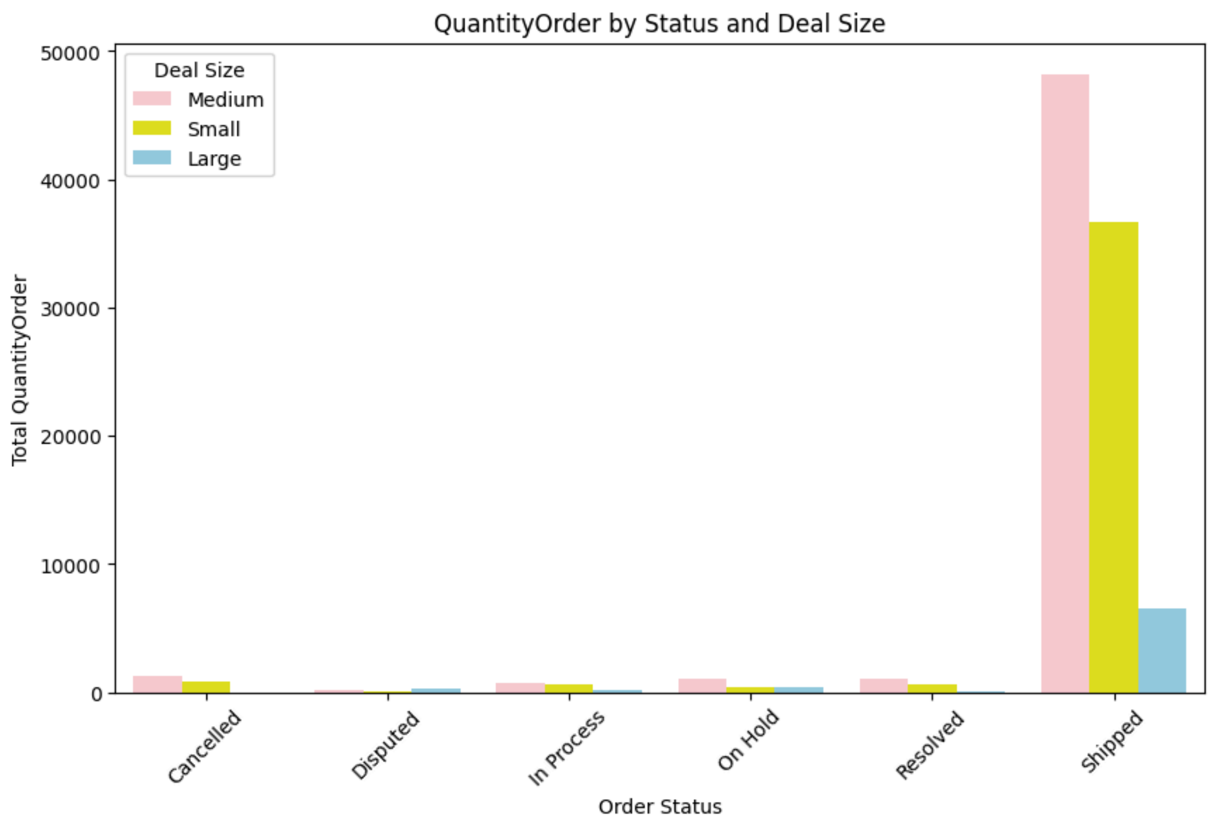
In [176…

```
status_dealsize_sales = df.groupby(['STATUS', 'DEALSIZE'])['SALES'].sum().reset_ind
plt.figure(figsize=(10, 6))
sns.barplot(x='STATUS', y='SALES', hue='DEALSIZE', data=status_dealsize_sales, pale

plt.xlabel('Order Status')
plt.ylabel('Total Sales')
plt.title('Sales by Status and Deal Size')
plt.xticks(rotation=45)
plt.legend(title='Deal Size')
plt.show()
```



In [201…
```
# In shipped mode medium dealsize performance is good
```

In [178…
```
status_dealsize_QuantityOrder = df.groupby(['STATUS', 'DEALSIZE'])['QUANTITYORDERED
plt.figure(figsize=(10, 6))
sns.barplot(x='STATUS', y='QUANTITYORDERED', hue='DEALSIZE', data=status_dealsize_Q

plt.xlabel('Order Status')
plt.ylabel('Total QuantityOrder')
plt.title('QuantityOrder by Status and Deal Size')
plt.xticks(rotation=45)
plt.legend(title='Deal Size')
plt.show()
```

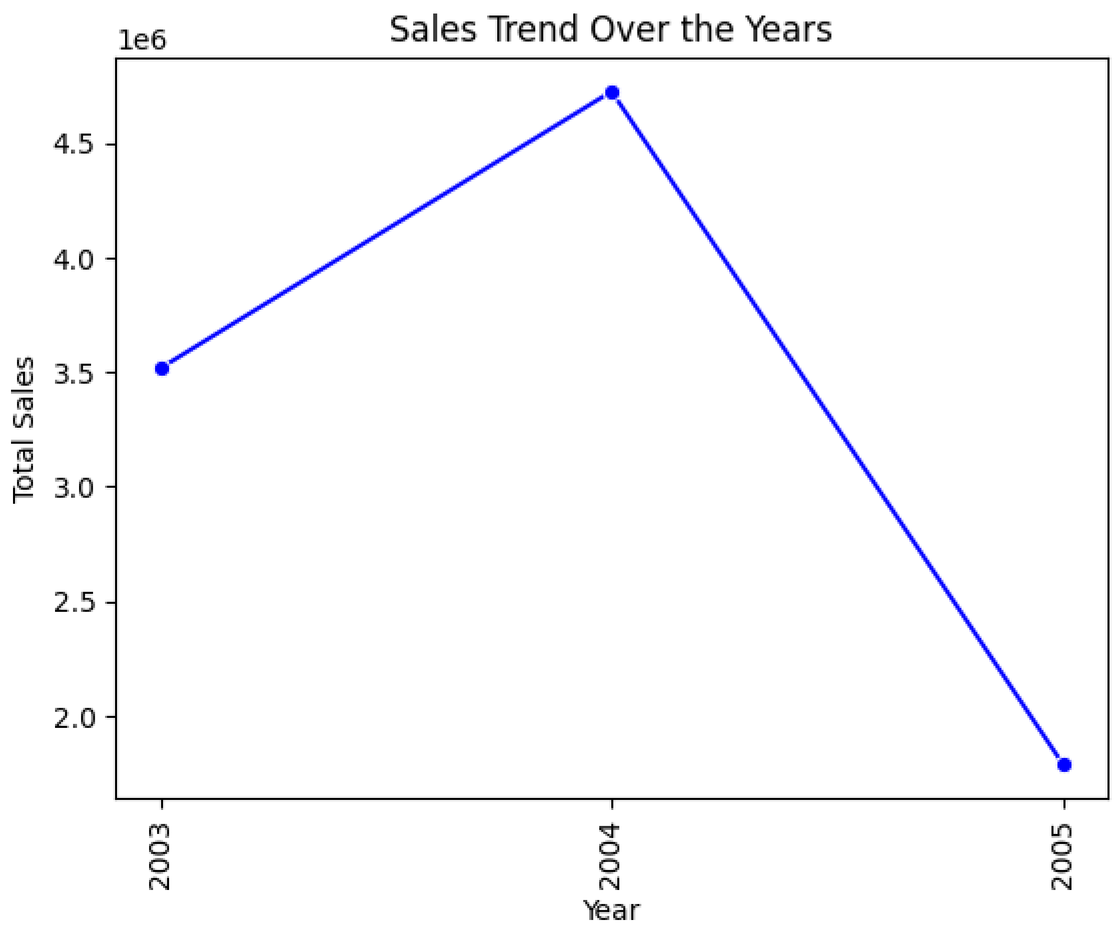


```
In [ ]:
```

Highlighting sales trends over time, seasonal patterns, and the top-performing products.

In [179…

```
yearly_sales = df.groupby('YEAR')['SALES'].sum().reset_index()
sns.lineplot(x=yearly_sales['YEAR'], y=yearly_sales['SALES'], marker='o', color='bl
plt.xlabel('Year')
plt.xticks(rotation=90)
plt.ylabel('Total Sales')
plt.title('Sales Trend Over the Years')
plt.show()
```

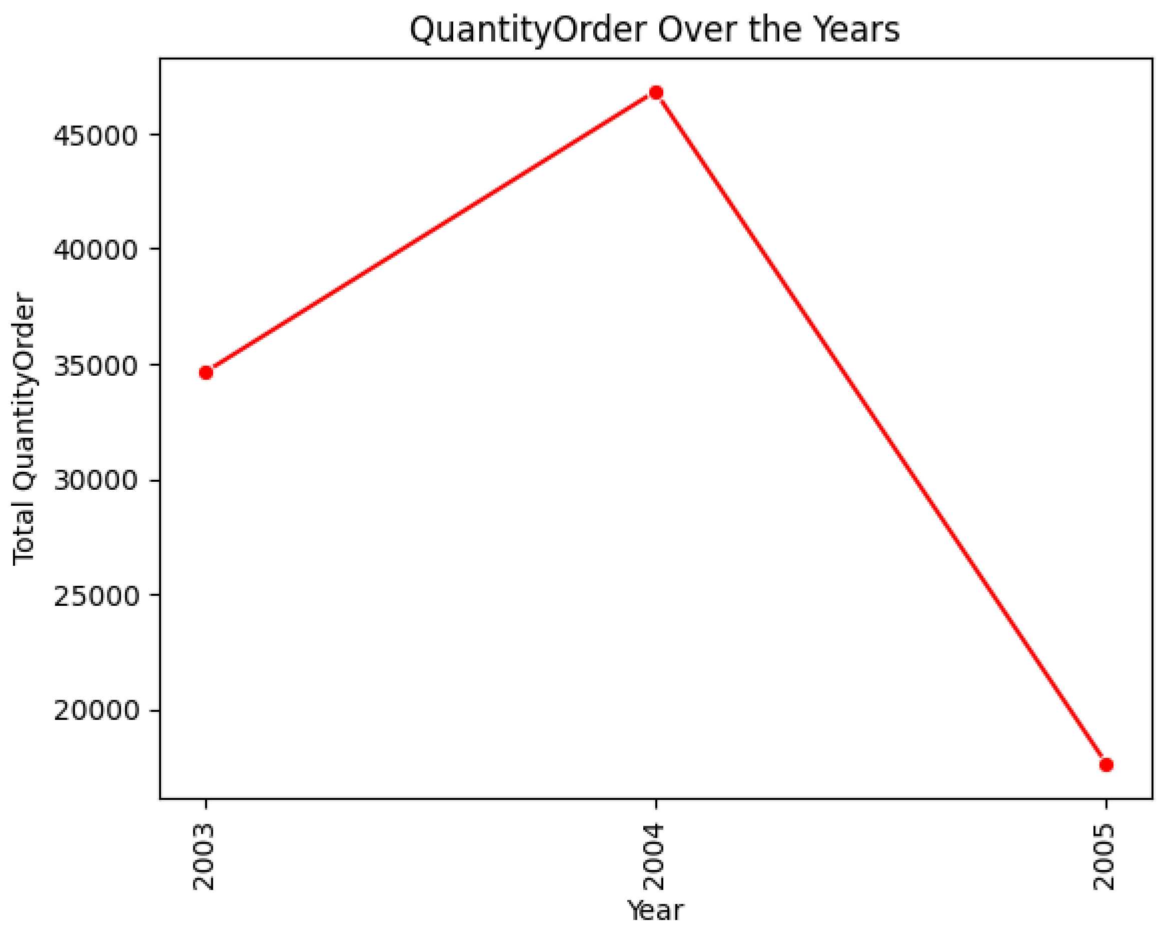


In [202…

```
# sales first increase after that it decrease overe year
```

In [189…

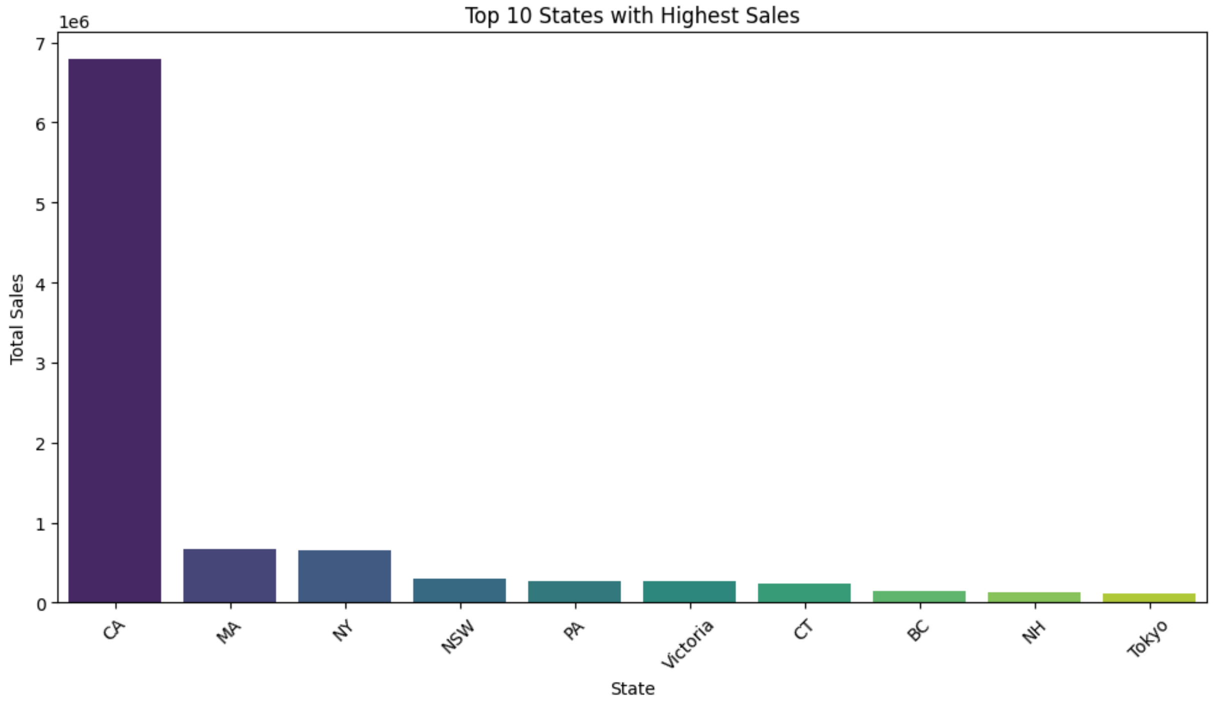
```
yearly_sales = df.groupby('YEAR')['QUANTITYORDERED'].sum().reset_index()
sns.lineplot(x=yearly_sales['YEAR'], y=yearly_sales['QUANTITYORDERED'], marker='o',
plt.xlabel('Year')
plt.xticks(rotation=90)
plt.ylabel('Total QuantityOrder')
plt.title('QuantityOrder Over the Years')
plt.show()
```



In [188…

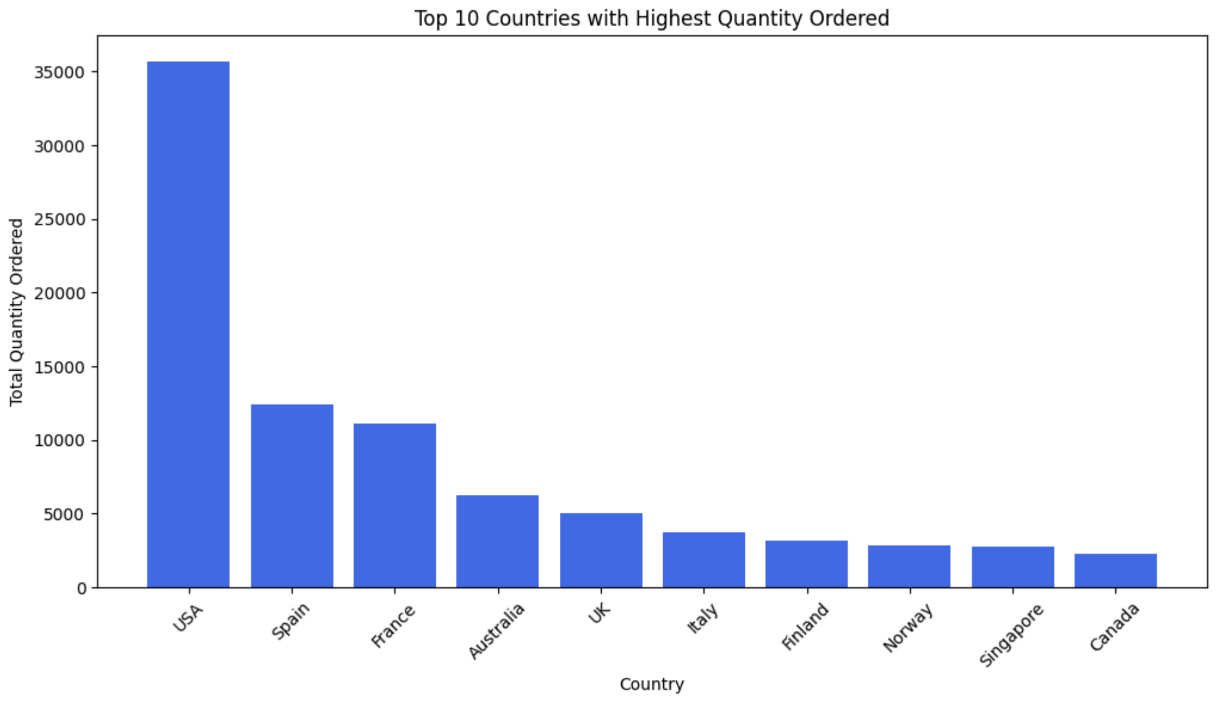
```
top_states = df.groupby('STATE')['SALES'].sum().nlargest(10).reset_index()
plt.figure(figsize=(12, 6))
sns.barplot(x='STATE', y='SALES', data=top_states, palette='viridis')

plt.xlabel('State')
plt.ylabel('Total Sales')
plt.title('Top 10 States with Highest Sales')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\ankit\AppData\Local\Temp\ipykernel_16820\3974003812.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='STATE', y='SALES', data=top_states, palette='viridis')
```



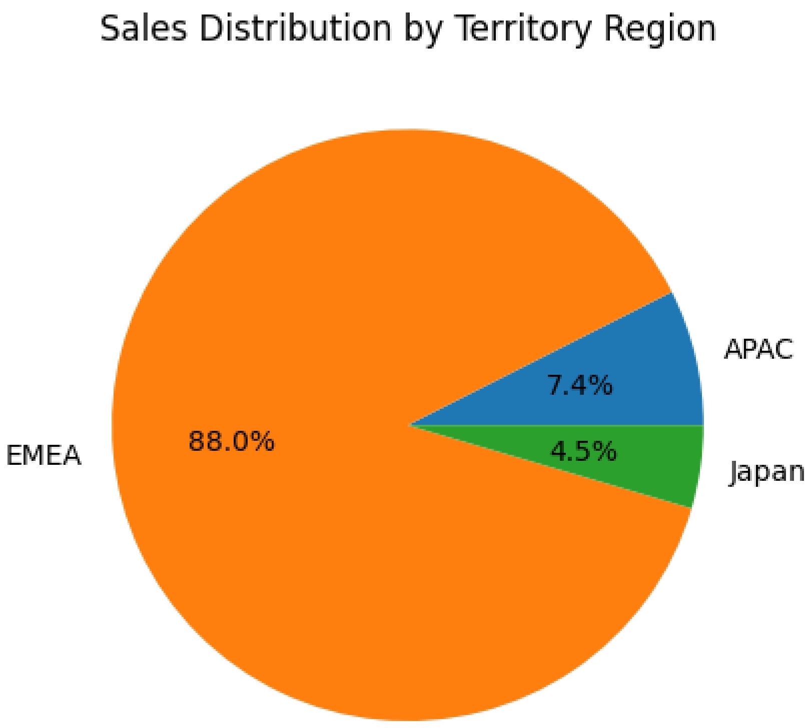
In [184…

```
top_countries = df.groupby('COUNTRY')['QUANTITYORDERED'].sum().nlargest(10)

plt.figure(figsize=(12, 6))
plt.bar(top_countries.index, top_countries.values, color='royalblue')

plt.xlabel('Country')
plt.ylabel('Total Quantity Ordered')
plt.title('Top 10 Countries with Highest Quantity Ordered')
plt.xticks(rotation=45)

plt.show()
```



In [197…

```
df.groupby('TERRITORY')['SALES'].sum().plot(kind='pie',autopct='%1.1f%%', title="Sa
plt.ylabel('')
plt.show()
```



In [ ]:

-- THANKYOU--------------------------------

In [ ]: