E-commerce Project on Sales Distribution

The following topics are analyzed and Visualization has been created by python module. Presented by Nasir Ahmed.

1. Sales Distribution
2. Yearly and Monthly Revenues
3. Discount Rates and Sales Relations
4. Number of Sales for Countries and Territories
5. Sales Distribution for Deal Sizes
6. Monthly Active Users
7. Sales Analysis by Product Line/Name
8. Sales Analysis according to Status
9. Monthly Profit Analysis
10. Profit Analysis by Deal Size
11. Profit Analysis by Product Line/Name
12. Sales and Profit Analysis by Deal Size
13. Analyse of Sales-to-Profit Ratio

In [1]:

```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import calendar
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
# Load CSV files into DataFrames
df = pd.read_csv("D:\\Python_PJ\\Sales\\sales_data.csv")
```

Understanding the data

In [3]:

```
df
```

Out[3]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	S
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	S
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	S
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	S
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	S
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	S
...	...	...	...	...	...	...	
2818	10350	20	100.00	15	2244.40	12/2/2004 0:00	S
2819	10373	29	100.00	1	3978.51	1/31/2005 0:00	S
2820	10386	43	100.00	4	5417.57	3/1/2005 0:00	Re
2821	10397	34	62.24	1	2116.16	3/28/2005 0:00	S
2822	10414	47	65.52	9	3079.44	5/6/2005 0:00	C

2823 rows × 25 columns

In [4]:

```
df.head()
```

Out[4]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STAT
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipp
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipp
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipp
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipp
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipp

5 rows × 25 columns

In [5]:
```
df.tail()
```

Out[5]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	S
2818	10350	20	100.00	15	2244.40	12/2/2004 0:00	S
2819	10373	29	100.00	1	3978.51	1/31/2005 0:00	S
2820	10386	43	100.00	4	5417.57	3/1/2005 0:00	R
2821	10397	34	62.24	1	2116.16	3/28/2005 0:00	S
2822	10414	47	65.52	9	3079.44	5/6/2005 0:00	C

5 rows × 25 columns

In [6]:
```
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
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['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')
```

Cleaning the data

In [8]:
```
# Dropping the redundant data, ie. removing the columns which are not necessary for analysis
df=df.drop(columns=["ORDERNUMBER","ORDERLINENUMBER","PRODUCTCODE","PHONE","ADDRESSLINE2",
               "STATE","POSTALCODE","CONTACTLASTNAME","CONTACTFIRSTNAME"])
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
QUANTITYORDERED       0
PRICEEACH             0
SALES                 0
ORDERDATE             0
STATUS                0
QTR_ID                0
MONTH_ID              0
YEAR_ID               0
PRODUCTLINE           0
MSRP                  0
CUSTOMERNAME          0
ADDRESSLINE1          0
CITY                  0
COUNTRY               0
TERRITORY          1074
DEALSIZE              0
dtype: int64
```

In [10]:
```
# Checking missing values
df.isnull().any()
```

Out[10]:
```
QUANTITYORDERED    False
PRICEEACH          False
SALES              False
ORDERDATE          False
STATUS             False
QTR_ID             False
MONTH_ID           False
YEAR_ID            False
PRODUCTLINE        False
MSRP               False
CUSTOMERNAME       False
ADDRESSLINE1       False
CITY               False
COUNTRY            False
TERRITORY           True
DEALSIZE           False
dtype: bool
```

In [11]:
```
df['TERRITORY'].unique()
```

Out[11]:
```
array([nan, 'EMEA', 'APAC', 'Japan'], dtype=object)
```

In [12]:
```
# Replacing the null value (i.e. NaN in the column 'TERRITORY')
df.fillna({'TERRITORY': 'N_AMERICA'}, inplace=True)
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
QUANTITYORDERED    0
PRICEEACH          0
SALES              0
ORDERDATE          0
STATUS             0
QTR_ID             0
MONTH_ID           0
YEAR_ID            0
PRODUCTLINE        0
MSRP               0
CUSTOMERNAME       0
ADDRESSLINE1       0
CITY               0
COUNTRY            0
TERRITORY          0
DEALSIZE           0
dtype: int64
```

In [14]:
```
df.nunique()
```

Out[14]:
```
QUANTITYORDERED      58
PRICEEACH          1016
SALES              2763
ORDERDATE           252
STATUS                6
QTR_ID                4
MONTH_ID             12
YEAR_ID               3
PRODUCTLINE           7
MSRP                 80
CUSTOMERNAME         92
ADDRESSLINE1         92
CITY                 73
COUNTRY              19
TERRITORY             4
DEALSIZE              3
dtype: int64
```

In [15]:
```
df['STATUS'].unique()
```

Out[15]:
```
array(['Shipped', 'Disputed', 'In Process', 'Cancelled', 'On Hold',
       'Resolved'], dtype=object)
```

In [16]:
```
df['DEALSIZE'].unique()
```

Out[16]:
```
array(['Small', 'Medium', 'Large'], dtype=object)
```

In [17]:
```
df.describe()
```

Out[17]:

	QUANTITYORDERED	PRICEEACH	SALES	QTR_ID	MONTH_ID	YEAR_ID	MS
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823.00000	2823.0000
mean	35.092809	83.658544	3553.889072	2.717676	7.092455	2003.81509	100.7155
std	9.741443	20.174277	1841.865106	1.203878	3.656633	0.69967	40.1879
min	6.000000	26.880000	482.130000	1.000000	1.000000	2003.00000	33.0000
25%	27.000000	68.860000	2203.430000	2.000000	4.000000	2003.00000	68.0000
50%	35.000000	95.700000	3184.800000	3.000000	8.000000	2004.00000	99.0000
75%	43.000000	100.000000	4508.000000	4.000000	11.000000	2004.00000	124.0000
max	97.000000	100.000000	14082.800000	4.000000	12.000000	2005.00000	214.0000

In [18]:

```
df.shape
```

Out[18]:

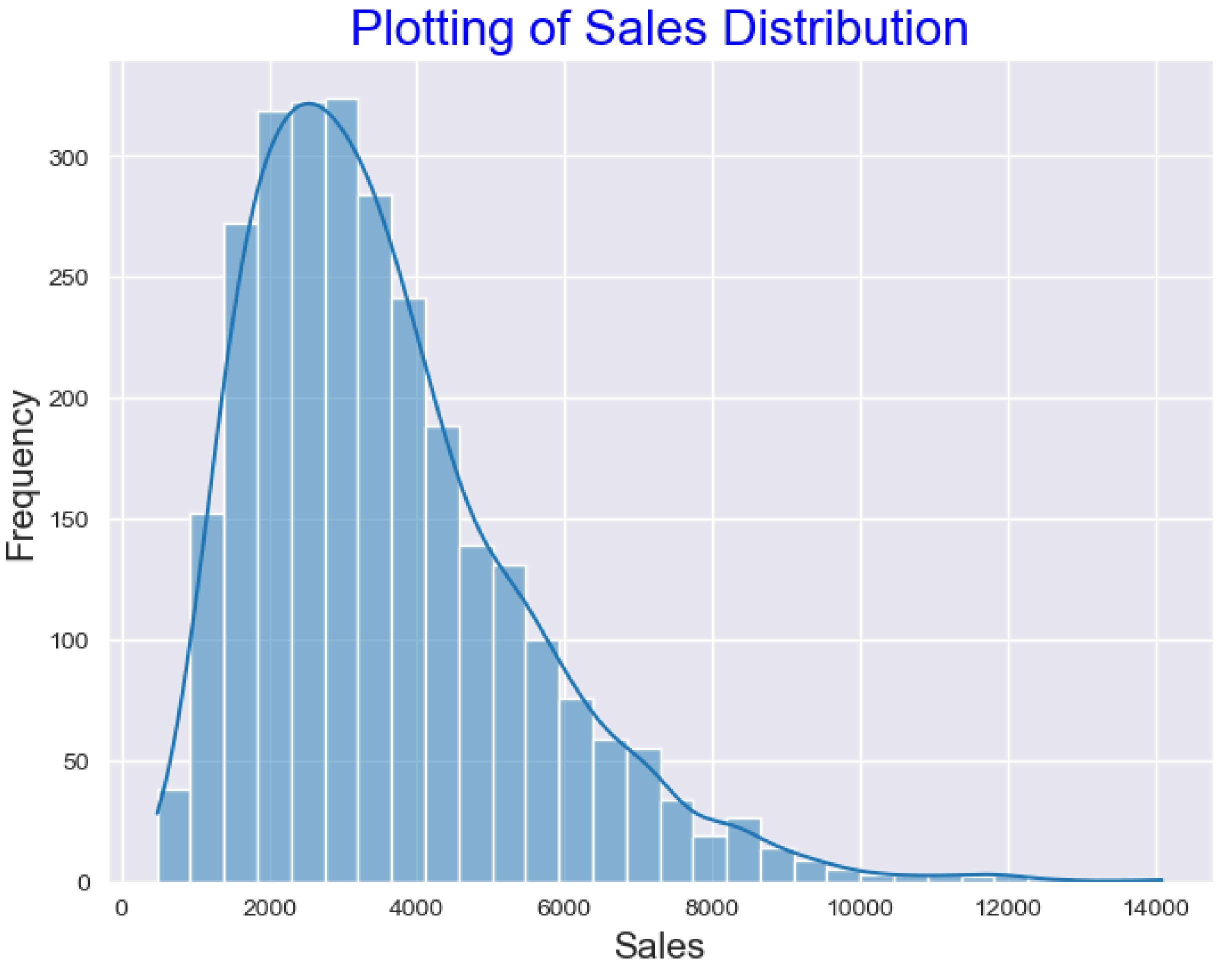
```
(2823, 16)
```

1. Sales Distribution

It is visualized in the histogram below. It shows the frequency of sales amounts across the dataset.

In [19]:

```
# Set the aesthetic style of the plots
sns.set_style("darkgrid")

# Sales Distribution
plt.figure(figsize=(8, 6))
sns.histplot(df['SALES'], bins=30, kde=True)
# plt.hist(df['SALES'], bins=30)
plt.title('Plotting of Sales Distribution', c="blue", size=20)
plt.xlabel('Sales', size=15)
plt.ylabel('Frequency', size=15)
plt.show()
```

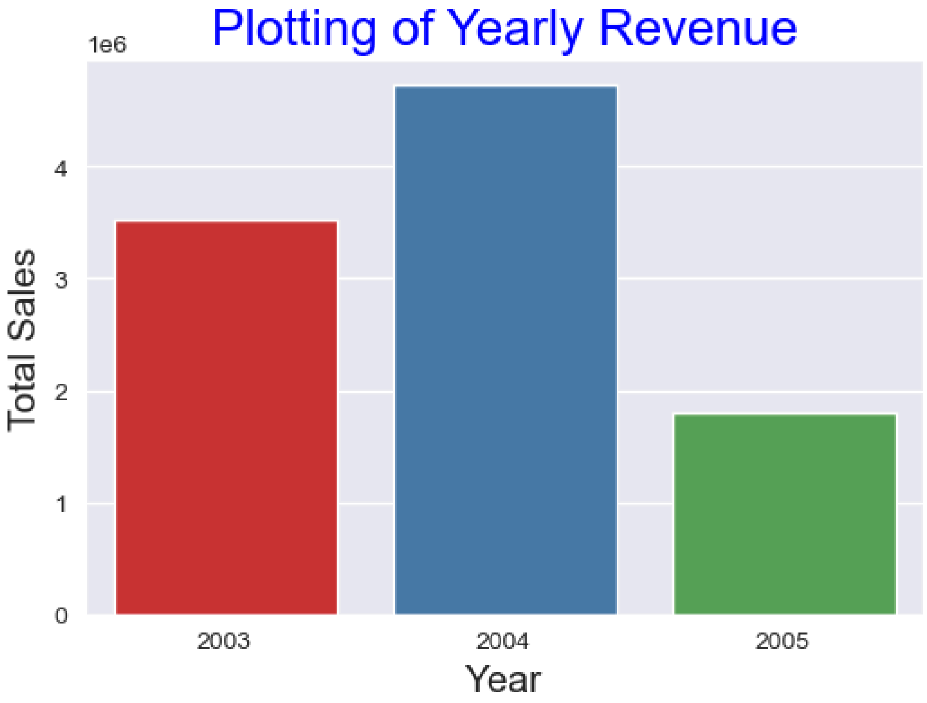


2. Analyzing of Yearly and Monthly Sales/Revenues

(a) Displaying the Yearly Revenues by Bar chart

In [20]:

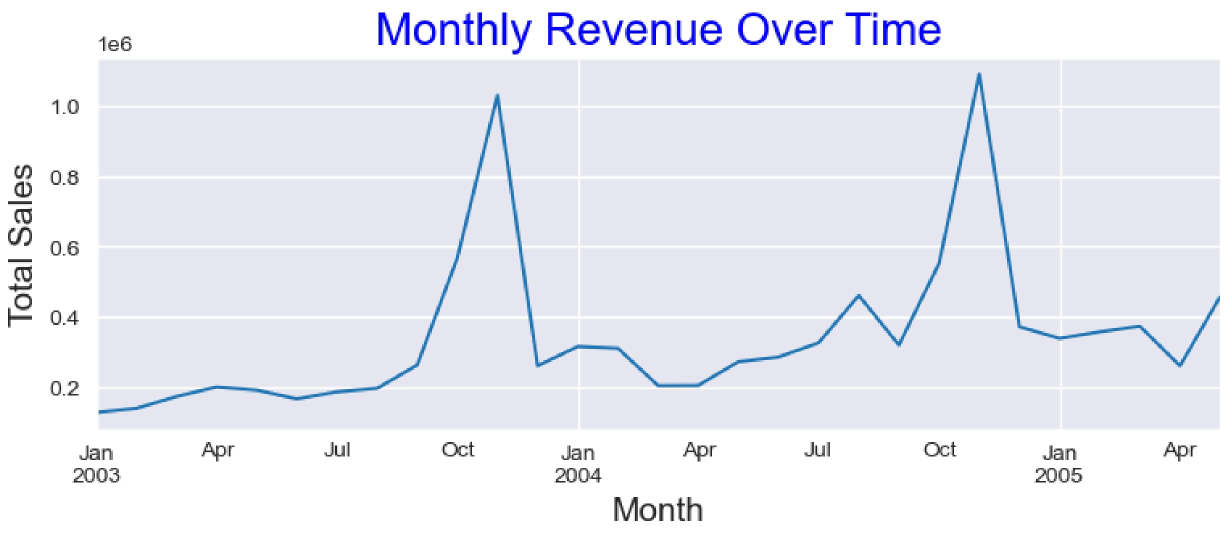
```
# Convert 'ORDERDATE' to datetime format
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])

# Aggregate sales data by year
yearly_revenue = df.groupby(df['ORDERDATE'].dt.year)['SALES'].sum().reset_index()
# Yearly Revenue
plt.figure(figsize=(6, 4), dpi=100)
#sns.barplot(x='YEAR_ID',y='SALES',data=df)
sns.barplot(x='ORDERDATE',y='SALES', data=yearly_revenue, palette='Set1', legend=False )

#yearly_revenue.plot(kind='bar')
plt.title('Plotting of Yearly Revenue', c="blue", size=20)
plt.xlabel('Year', size=15)
plt.ylabel('Total Sales', size=15)
plt.xticks(rotation=0)
plt.show()
```



(b) Displaying the Monthly Revenues by Line chart

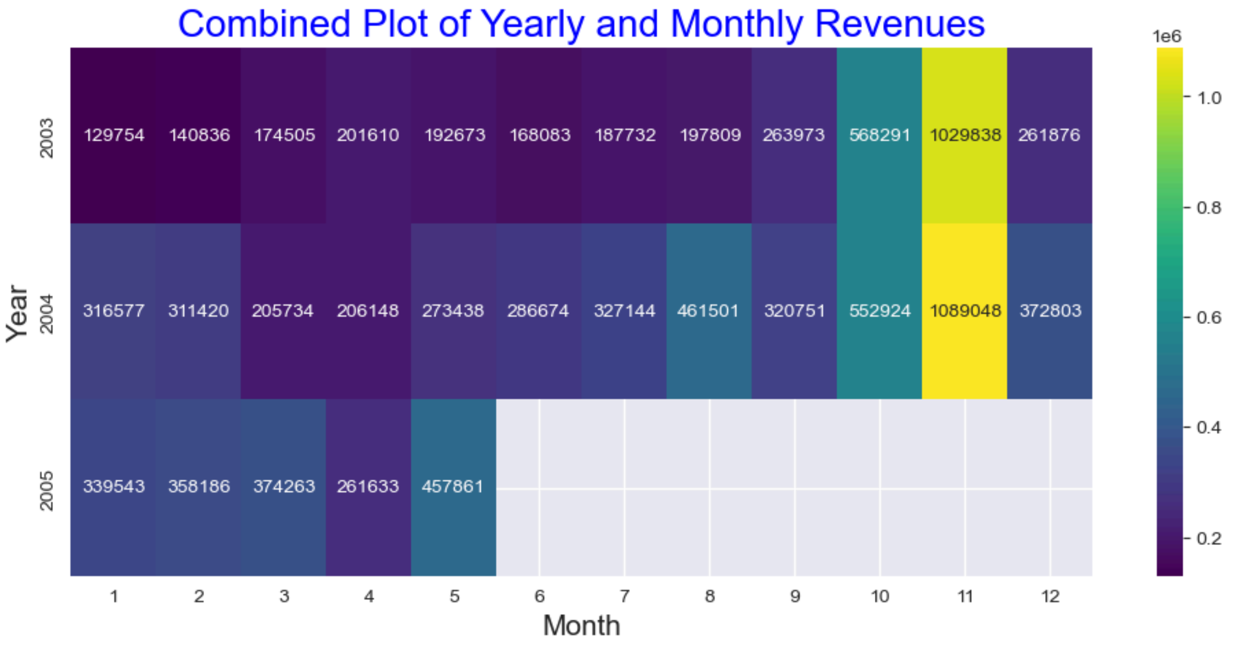
In [21]:

```
# Convert 'ORDERDATE' to datetime format
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
df['MONTHYEAR'] = df['ORDERDATE'].dt.to_period('M')

# Aggregate data by year
monthly_revenue = df.groupby('MONTHYEAR')['SALES'].sum()

# Monthly Revenue
plt.figure(figsize=(9, 3))
monthly_revenue.plot(kind='line')
plt.title('Monthly Revenue Over Time', c="blue", size=20)
plt.xlabel('Month', size=15)
plt.ylabel('Total Sales', size=15)
plt.xticks(rotation=0)
plt.show()
```



(c) Combined or comparison plot of Yearly and Monthly Revenues

In [22]:
```
plt.figure(figsize=(12, 5))

# Grouping data by year and month
monthly_revenue = df.groupby(['YEAR_ID', 'MONTH_ID'])['SALES'].sum().reset_index()

# Pivot table to have months on the columns and years on the rows
monthly_revenue_pivot = monthly_revenue.pivot_table(index='YEAR_ID', columns='MONTH_ID', value

# Plotting the heatmap
sns.heatmap(monthly_revenue_pivot, cmap='viridis', annot=True, fmt='.0f')
plt.title('Combined Plot of Yearly and Monthly Revenues', c="blue", size=20)
plt.xlabel('Month', size=15)
plt.ylabel('Year', size=15)
plt.show()
```



3. Discount Rates and Sales Relations

A relationship between Discount Rates and Sales has been analyzed.

- Discount Rate is calculated as; DISCOUNTRATE = (MSRP - PRICEEACH)/MSRP*100
- Where, MSRP = Manufacturer's Suggested Retail Price
- and PRICEEACH = Manufacturing Cost of Each Product

The scatter plot below illustrates the relationship between Discount Rates and Sales. Each point represents a sale with its corresponding discount rate.

In [23]:

```
df['DISCOUNT_RATE'] = ((df['MSRP']-df['PRICEEACH'])/df['MSRP'])*100
print(df[['PRICEEACH','MSRP','DISCOUNT_RATE']])
```

```
      PRICEEACH  MSRP  DISCOUNT_RATE
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
...         ...   ...            ...
2818     100.00    54     -85.185185
2819     100.00    54     -85.185185
2820     100.00    54     -85.185185
2821      62.24    54     -15.259259
2822      65.52    54     -21.333333

[2823 rows x 3 columns]
```

In [24]:

```
df.columns
```

Out[24]:

```
Index(['QUANTITYORDERED', 'PRICEEACH', 'SALES', 'ORDERDATE', 'STATUS',
       'QTR_ID', 'MONTH_ID', 'YEAR_ID', 'PRODUCTLINE', 'MSRP', 'CUSTOMERNAME',
       'ADDRESSLINE1', 'CITY', 'COUNTRY', 'TERRITORY', 'DEALSIZE', 'MONTHYEAR',
       'DISCOUNT_RATE'],
      dtype='object')
```

In [25]:

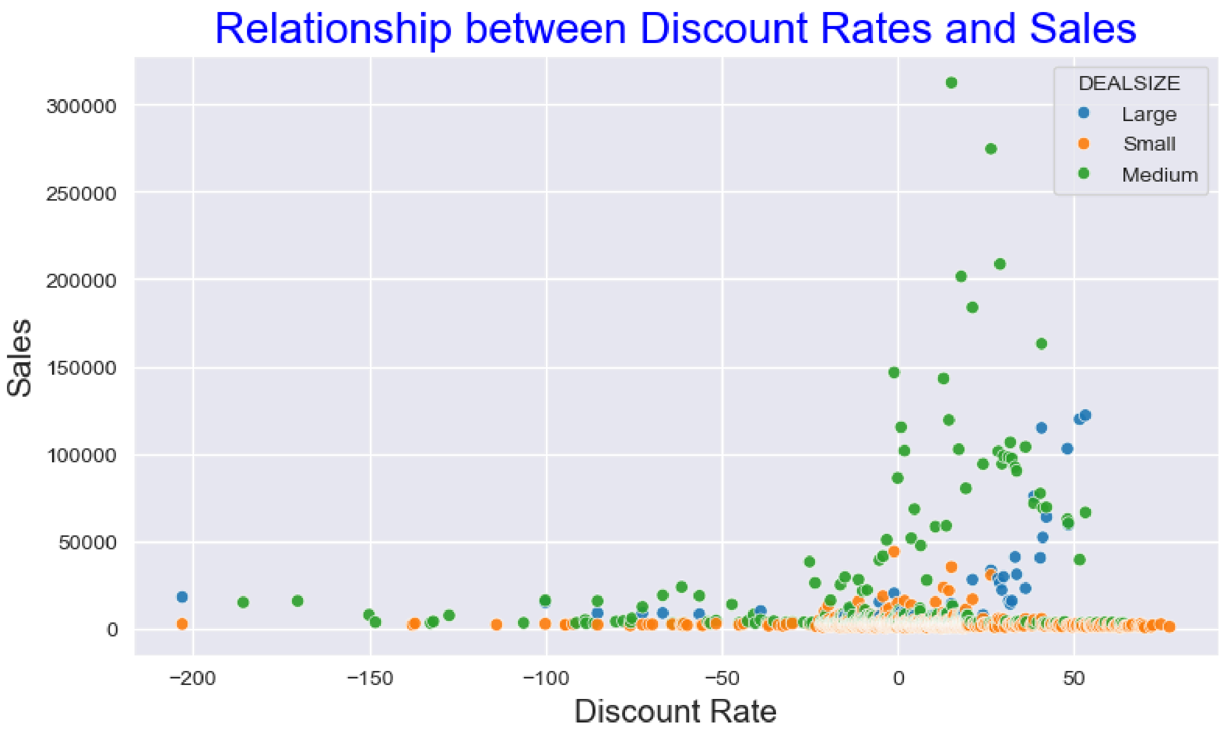
```
df.shape
```

Out[25]:

```
(2823, 18)
```

In [26]:

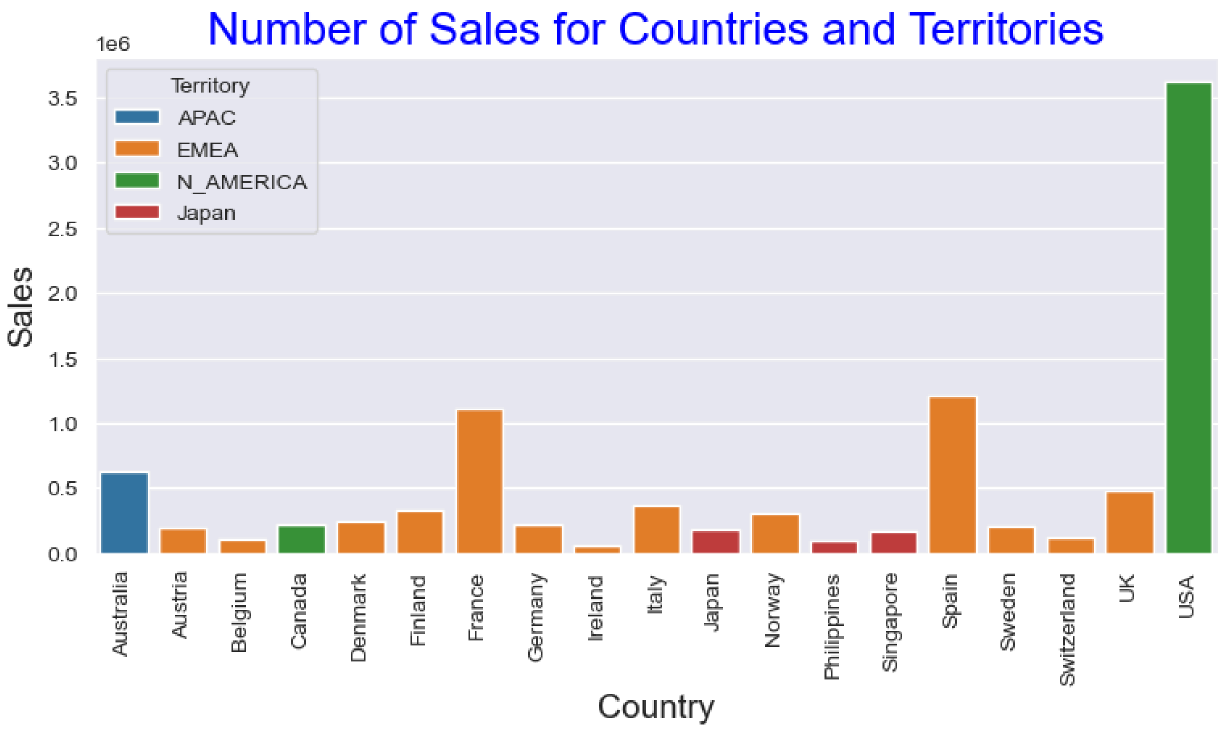
```
plt.figure(figsize=(9, 5))

# Grouping data by discount rate and dealsize
sales_by_discountrate = df.groupby(['DISCOUNT_RATE','DEALSIZE'])['SALES'].sum().reset_index()
sns.scatterplot(data=sales_by_discountrate, x='DISCOUNT_RATE', y='SALES', hue='DEALSIZE', alph
plt.title('Relationship between Discount Rates and Sales', c="blue", size=20)
plt.xlabel('Discount Rate', size=15)
plt.ylabel('Sales', size=15)
plt.show()
```



4. Number of Sales for Countries and Territories

The bar plot below shows the Number of Sales for each Country, segmented by Territories. The length of the bars represents the count of sales.

In [27]:
```
plt.figure(figsize=(9, 4))

# Grouping data by country and territory
sales_by_location = df.groupby(['COUNTRY', 'TERRITORY'])['SALES'].sum().reset_index()

# Creating a bar plot
sns.barplot(data=sales_by_location, x='COUNTRY', y='SALES', hue='TERRITORY', dodge=False)
plt.title('Number of Sales for Countries and Territories', c="blue", size=20)
plt.xlabel('Country', size=15)
plt.ylabel('Sales', size=15)
plt.xticks(rotation=90)
plt.legend(title='Territory')
plt.show()
```



5. Sales Distribution with respect to Deal Sizes.

The pie chart below represents the Sales Distribution for different Deal Sizes. Each slice of the pie corresponds to the proportion of total sales for a particular deal size.

In [28]:

```
# Creating a donut chart using Plotly Express
# plotly.express (px): Data visualization library for making quick plots.
import plotly.express as px

# Grouping data by deal size
deal_size_sales = df.groupby('DEALSIZE')['SALES'].sum().reset_index()

fig = px.pie(deal_size_sales,
             values='SALES',
             names='DEALSIZE',
             hole=0.2,
             color_discrete_sequence=px.colors.qualitative.Dark2)

# Updating the layout
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title_text='Pie Chart for Sales Distribution w.r.t Deal Sizes', title_font=d
fig.show()
```

6. Monthly Active Users

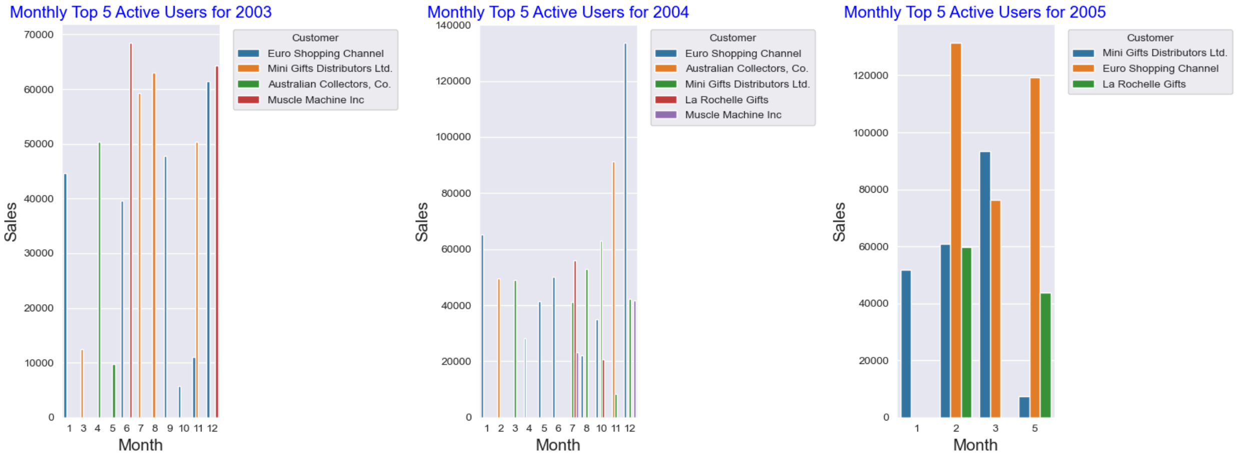
- Active Users mean ‘Customers' Names’. The column name is 'CUSTOMERNAME'.

Monthly sales report based on the top five customers

In [29]:

```
# Convert ORDERDATE to datetime
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])

# Group by YEAR_ID, MONTH_ID, and CUSTOMERNAME and sum the SALES
monthly_sales = df.groupby(['YEAR_ID', 'MONTH_ID', 'CUSTOMERNAME'])['SALES'].sum().reset_index

# Identify the top five customers based on total sales
top_customers = monthly_sales.groupby('CUSTOMERNAME')['SALES'].sum().nlargest(5).index.tolist(

# Filter the data for only the top five customers
top_customers_sales = monthly_sales[monthly_sales['CUSTOMERNAME'].isin(top_customers)]
'top_customers_sales' in locals()

# Filter the data for the years 2003, 2004, and 2005
filtered_data = top_customers_sales[top_customers_sales['YEAR_ID'].isin([2003, 2004, 2005])]

# Set the aesthetic style of the plots
```

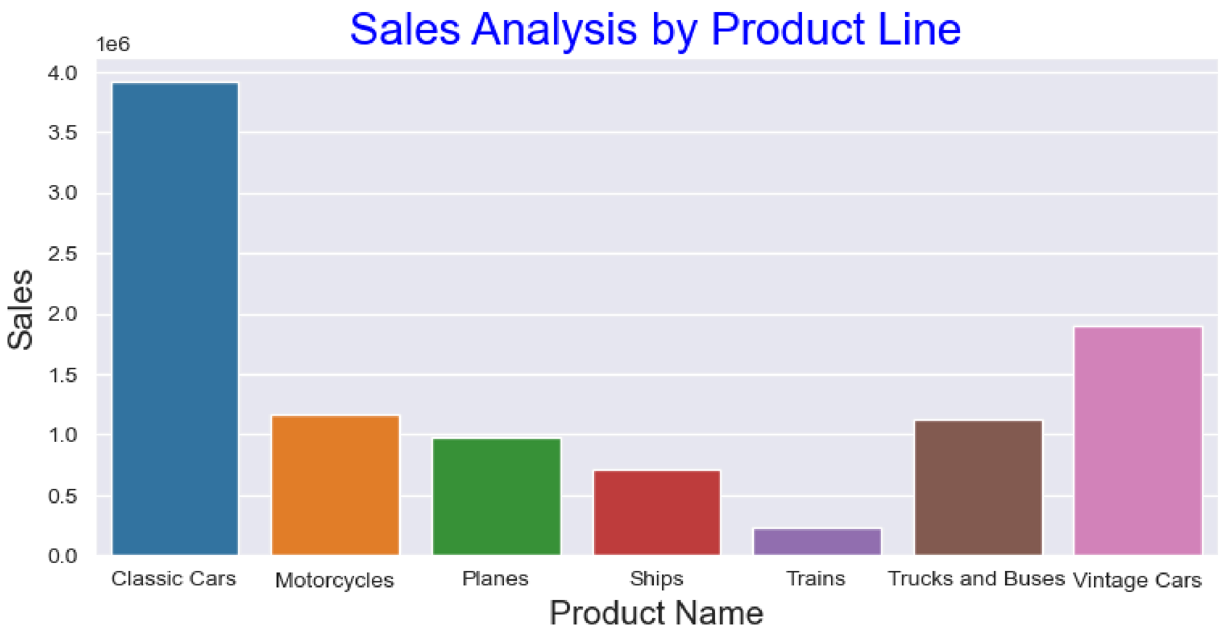
```
sns.set_style('darkgrid')

# Initialize the figure
plt.figure(figsize=(16, 6))

# Plot bar chart for each year
for i, year in enumerate([2003, 2004, 2005], 1):
    plt.subplot(1, 3, i)
    year_data = filtered_data[filtered_data['YEAR_ID'] == year]
    sns.barplot(data=year_data, x='MONTH_ID', y='SALES', hue='CUSTOMERNAME', ci=None)
    plt.title('Monthly Top 5 Active Users for ' + str(year), c="blue", size=15)
    plt.xlabel('Month', size=15)
    plt.ylabel('Sales', size=15)
    plt.legend(title='Customer', bbox_to_anchor=(1.05, 1), loc='upper left')

# Adjust the layout and show the plot
plt.tight_layout()
plt.show()
```



7. Sales Analysis by Product Line/Name

In [30]:

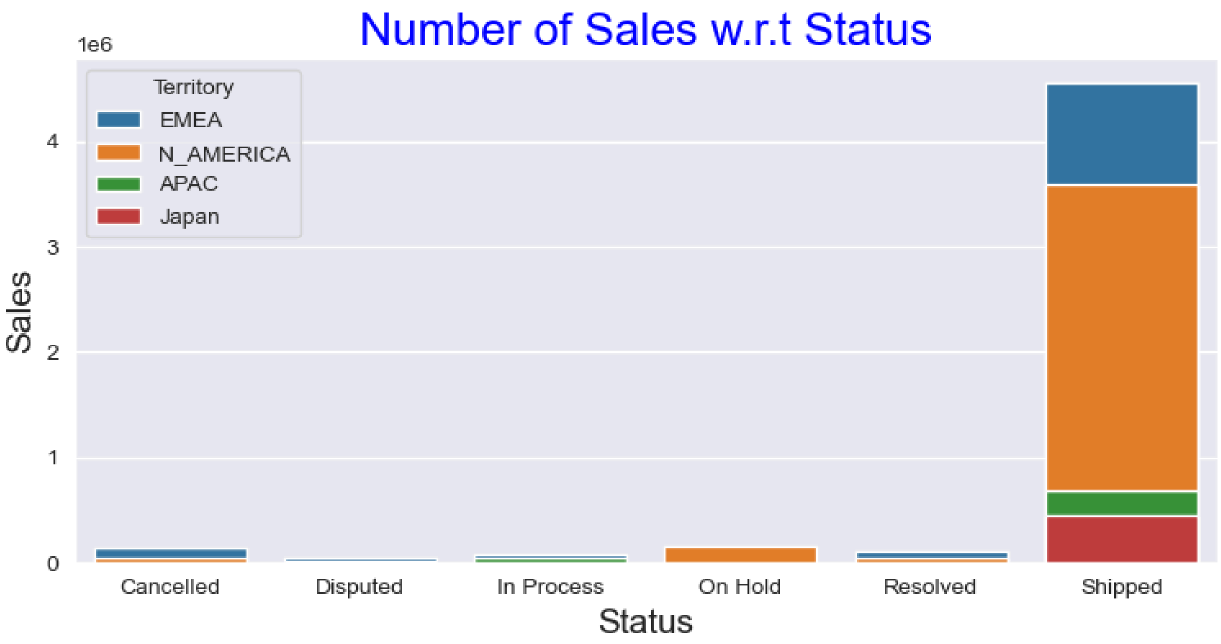
```
# Plotting of Sales according to Products
plt.figure(figsize=(9, 4))

# Grouping data by country and territory
sales_by_productline = df.groupby('PRODUCTLINE')['SALES'].sum().reset_index()

# Creating a bar plot
sns.barplot(data=sales_by_productline, x='PRODUCTLINE', y='SALES', hue='PRODUCTLINE', dodge=Fa
plt.title('Sales Analysis by Product Line', c="blue", size=20)
plt.xlabel('Product Name', size=15)
plt.ylabel('Sales', size=15)
plt.xticks(rotation=0)
plt.show()
```



8. Sales Analysis according to Status

In [31]:

```
plt.figure(figsize=(9, 4))
# Grouping data by status and territory
sales_by_status = df.groupby(['STATUS', 'TERRITORY'])['SALES'].sum().reset_index()

# Creating a bar plot
sns.barplot(data=sales_by_status, x='STATUS', y='SALES', hue='TERRITORY', dodge=False)
plt.title('Number of Sales w.r.t Status', c="blue", size=20)
plt.xlabel('Status', size=15)
plt.ylabel('Sales', size=15)
plt.xticks(rotation=0)
plt.legend(title='Territory')
plt.show()
```

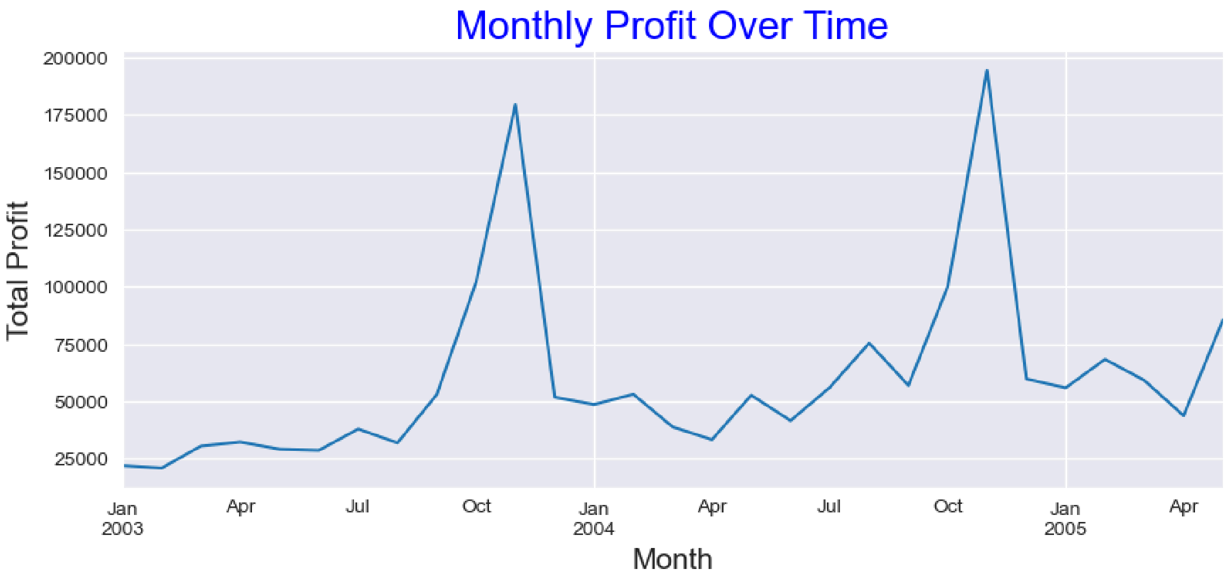


9. Monthly Profit Analysis

In [32]:
```
# Calculating PRODUCT COST and PROFIT
df['PRODUCT_COST'] = (df['QUANTITYORDERED']*df['PRICEEACH'])
df['PROFIT'] = (df['SALES']-df['PRODUCT_COST'])

# Displaying the relevant columns
print(df[['QUANTITYORDERED','PRICEEACH','PRODUCT_COST','PROFIT']].head())
```

```
   QUANTITYORDERED  PRICEEACH  PRODUCT_COST        PROFIT
0               30      95.70       2871.00  0.000000e+00
1               34      81.35       2765.90  4.547474e-13
2               41      94.74       3884.34  4.547474e-13
3               45      83.26       3746.70 -4.547474e-13
4               49     100.00       4900.00  3.052700e+02
```

In [33]:
```
# Convert 'ORDERDATE' to datetime format
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
df['MONTHYEAR'] = df['ORDERDATE'].dt.to_period('M')

# Aggregate data by year
monthly_profit = df.groupby('MONTHYEAR')['PROFIT'].sum()

# Monthly Profit
plt.figure(figsize=(10, 4))
monthly_profit.plot(kind='line')
plt.title('Monthly Profit Over Time', c="blue", size=20)
plt.xlabel('Month', size=15)
plt.ylabel('Total Profit', size=15)
plt.xticks(rotation=0)
plt.show()
```



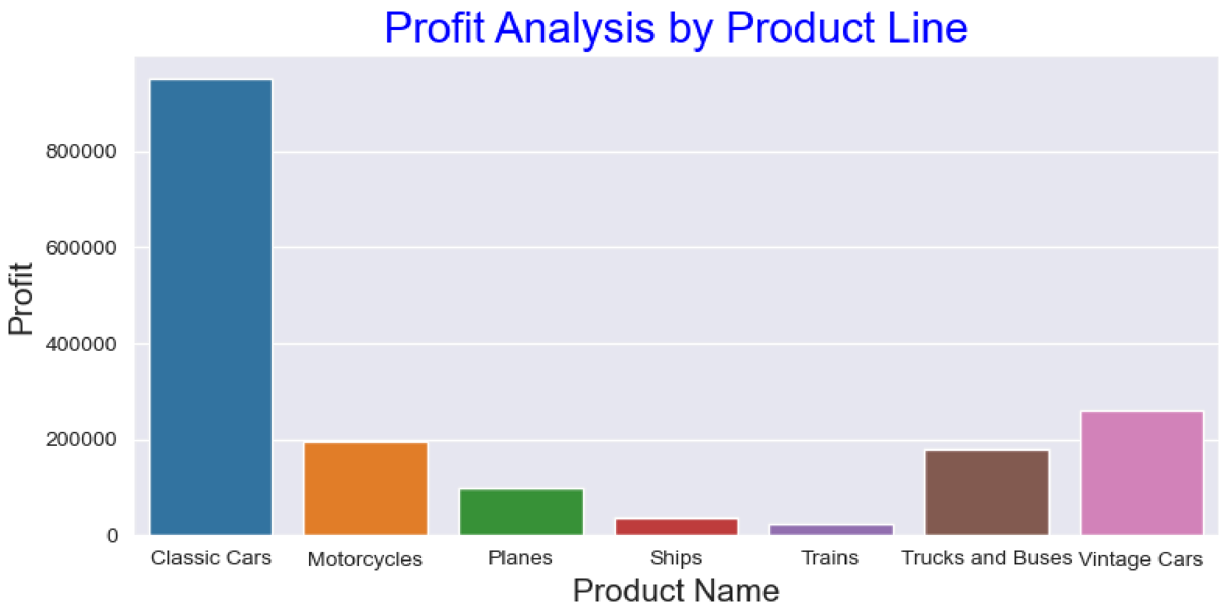
10. Profit Analysis by Deal Size

In [34]:
```
# Grouping data by deal size
profit_by_dealsize = df.groupby('DEALSIZE')['PROFIT'].sum().reset_index()

fig = px.pie(profit_by_dealsize,
             values='PROFIT',
             names='DEALSIZE',
             hole=0.2,
             color_discrete_sequence=px.colors.qualitative.Pastel)

# Updating the layout
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title_text='Profit Analysis by Deal Size', title_font=dict(size=24), width=7
fig.show()
```

11. Profit Analysis by Product Line/Name

In [35]:

```
# Plotting of Profit according to Products
plt.figure(figsize=(9, 4))

# Grouping data by country and territory
profit_by_productline = df.groupby('PRODUCTLINE')['PROFIT'].sum().reset_index()

# Creating a bar plot
sns.barplot(data=profit_by_productline, x='PRODUCTLINE', y='PROFIT', hue='PRODUCTLINE', dodge=
plt.title('Profit Analysis by Product Line', c="blue", size=20)
plt.xlabel('Product Name', size=15)
plt.ylabel('Profit', size=15)
plt.xticks(rotation=0)
plt.show()
```



12. Sales and Profit Analysis by Deal Size

```
In [36]: import plotly.graph_objects as go    #plotly.graph_objects (go): For making Advanced and custo
         import plotly.colors as colors

         sales_profit_by_dsize = df.groupby('DEALSIZE').agg({'SALES': 'sum', 'PROFIT': 'sum'}).reset_in
         color_palette = colors.qualitative.Dark2

         fig = go.Figure()
         fig.add_trace(go.Bar(x=sales_profit_by_dsize['DEALSIZE'],
                              y=sales_profit_by_dsize['SALES'],
                              name='Sales',
                              marker_color=color_palette[0]))

         fig.add_trace(go.Bar(x=sales_profit_by_dsize['DEALSIZE'],
                              y=sales_profit_by_dsize['PROFIT'],
                              name='Profit',
                              marker_color=color_palette[1]))

         fig.update_layout(
              title='Sales and Profit Analysis by Deal Size',
              xaxis_title='Customer Deal Size',
              yaxis_title='Sales',
              width=800,  # Width of the figure in pixels
              height=450  # Height of the figure in pixels
          )

         fig.show()
```

13. Analyse of Sales-to-Profit Ratio

In [37]:

```
sales_profit_by_dsize = df.groupby('DEALSIZE').agg({'SALES': 'sum', 'PROFIT': 'sum'}).reset_in
sales_profit_by_dsize['Sales_to_Profit_Ratio'] = sales_profit_by_dsize['SALES'] / sales_profit
print(sales_profit_by_dsize[['DEALSIZE', 'Sales_to_Profit_Ratio']])
```

```
  DEALSIZE  Sales_to_Profit_Ratio
0    Large               2.311339
1   Medium               5.407707
2    Small              50.167407
```

In [ ]: