Superstore Sales Analysis using Python

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as colors

```

In [2]:
```
#Load the data from CSV file
# encoding = 'latin-1' this is particularly useful if the file contains special characters not represented
#                      in the default encoding(usually UTF-8).
# df.head() Display the first few rows of the dataset
df = pd.read_csv(r"F:\kailash\Project\python superstore sale\superstore sale dataset.csv", encoding = 'latin-1')
print(df.head())

```

```
   Row ID        Order ID  Order Date   Ship Date       Ship Mode Customer ID  \
0       1  CA-2016-152156  11-08-2016  11-11-2016    Second Class    CG-12520
1       2  CA-2016-152156  11-08-2016  11-11-2016    Second Class    CG-12520
2       3  CA-2016-138688  06-12-2016   6/16/2016    Second Class    DV-13045
3       4  US-2015-108966  10-11-2015  10/18/2015  Standard Class    SO-20335
4       5  US-2015-108966  10-11-2015  10/18/2015  Standard Class    SO-20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...

  Postal Code  Region       Product ID         Category Sub-Category  \
0       42420   South  FUR-BO-10001798        Furniture    Bookcases
1       42420   South  FUR-CH-10000454        Furniture       Chairs
2       90036    West  OFF-LA-10000240  Office Supplies       Labels
3       33311   South  FUR-TA-10000577        Furniture       Tables
4       33311   South  OFF-ST-10000760  Office Supplies      Storage

                                     Product Name     Sales  Quantity  \
0                  Bush Somerset Collection Bookcase  261.9600         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.9400         3
2  Self-Adhesive Address Labels for Typewriters b...   14.6200         2
3      Bretford CR4500 Series Slim Rectangular Table  957.5775         5
4                    Eldon Fold 'N Roll Cart System   22.3680         2

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.00    6.8714
3      0.45 -383.0310
4      0.20    2.5164

[5 rows x 21 columns]
```

In [3]:
```
# Display Summary statistics
df.describe()

```

Out[3]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28.656896
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29.364000
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399.976000

In [4]:
```
# Display All Columns name
df.columns

```

Out[4]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [5]:
```
# Display basic information about the dataset
df.info()

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row ID         9994 non-null   int64  
 1   Order ID       9994 non-null   object 
 2   Order Date     9994 non-null   object 
 3   Ship Date      9994 non-null   object 
 4   Ship Mode      9994 non-null   object 
 5   Customer ID    9994 non-null   object 
 6   Customer Name  9994 non-null   object 
 7   Segment        9994 non-null   object 
 8   Country        9994 non-null   object 
 9   City           9994 non-null   object 
 10  State          9994 non-null   object 
 11  Postal Code    9994 non-null   int64  
 12  Region         9994 non-null   object 
 13  Product ID     9994 non-null   object 
 14  Category       9994 non-null   object 
 15  Sub-Category   9994 non-null   object 
 16  Product Name   9994 non-null   object 
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64  
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

In [6]:
```
# Display the number of unique values in each column
df.nunique()

```

Out[6]:
```
Row ID          9994
Order ID        5009
Order Date      1237
Ship Date       1334
Ship Mode          4
Customer ID      793
Customer Name    793
Segment            3
Country            1
City             531
State             49
Postal Code      631
Region             4
Product ID      1862
Category           3
Sub-Category      17
Product Name    1850
Sales           5825
Quantity          14
Discount          12
Profit          7287
dtype: int64
```

In [7]:
```
#Create new column like order month,order year,order day

# Convert the 'Order Date' column to datetime format
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
# Create new columns for order month, year, and day
df['Order Day'] = df['Order Date'].dt.day
df['Order Month'] = df['Order Date'].dt.month
df['Order Year'] = df['Order Date'].dt.year


df['Ship Day'] = df['Ship Date'].dt.day
df['Ship Month'] = df['Ship Date'].dt.month
df['Ship Year'] = df['Ship Date'].dt.year


# Display the updated DataFrame
print(df[['Order Date', 'Order Day', 'Order Month', 'Order Year',
            'Ship Date', 'Ship Day', 'Ship Month', 'Ship Year']])


```

```
     Order Date  Order Day  Order Month  Order Year  Ship Date  Ship Day  \
0    2016-11-08          8           11        2016 2016-11-11        11
1    2016-11-08          8           11        2016 2016-11-11        11
2    2016-06-12         12            6        2016 2016-06-16        16
3    2015-10-11         11           10        2015 2015-10-18        18
4    2015-10-11         11           10        2015 2015-10-18        18
...         ...        ...          ...         ...        ...       ...
9989 2014-01-21         21            1        2014 2014-01-23        23
9990 2017-02-26         26            2        2017 2017-03-03         3
9991 2017-02-26         26            2        2017 2017-03-03         3
9992 2017-02-26         26            2        2017 2017-03-03         3
9993 2017-05-04          4            5        2017 2017-05-09         9

      Ship Month  Ship Year
0             11       2016
1             11       2016
2              6       2016
3             10       2015
4             10       2015
...          ...        ...
9989           1       2014
9990           3       2017
9991           3       2017
9992           3       2017
9993           5       2017

[9994 rows x 8 columns]
```

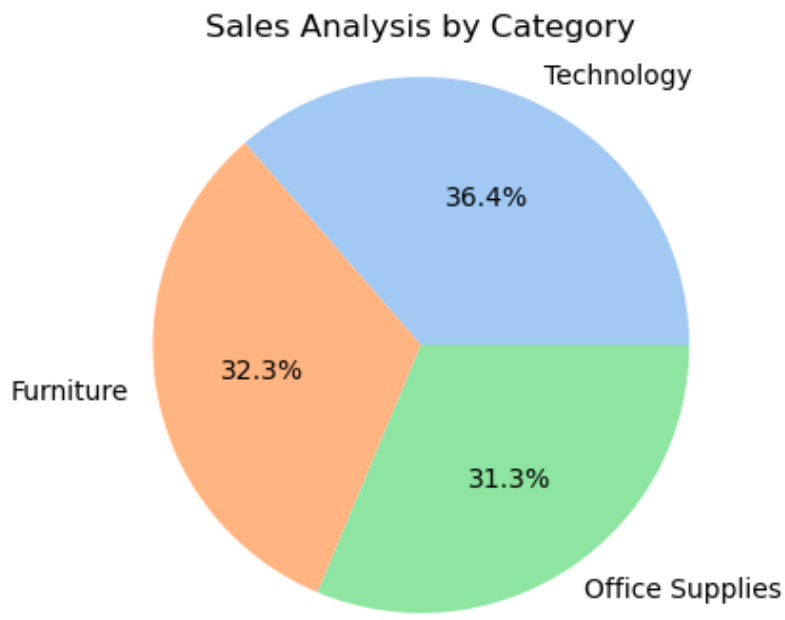
.

Before I start,I want to tell you.I have Created Two graph of each analysis,first using Matplotlib.pyplot library and second using plotly.express library

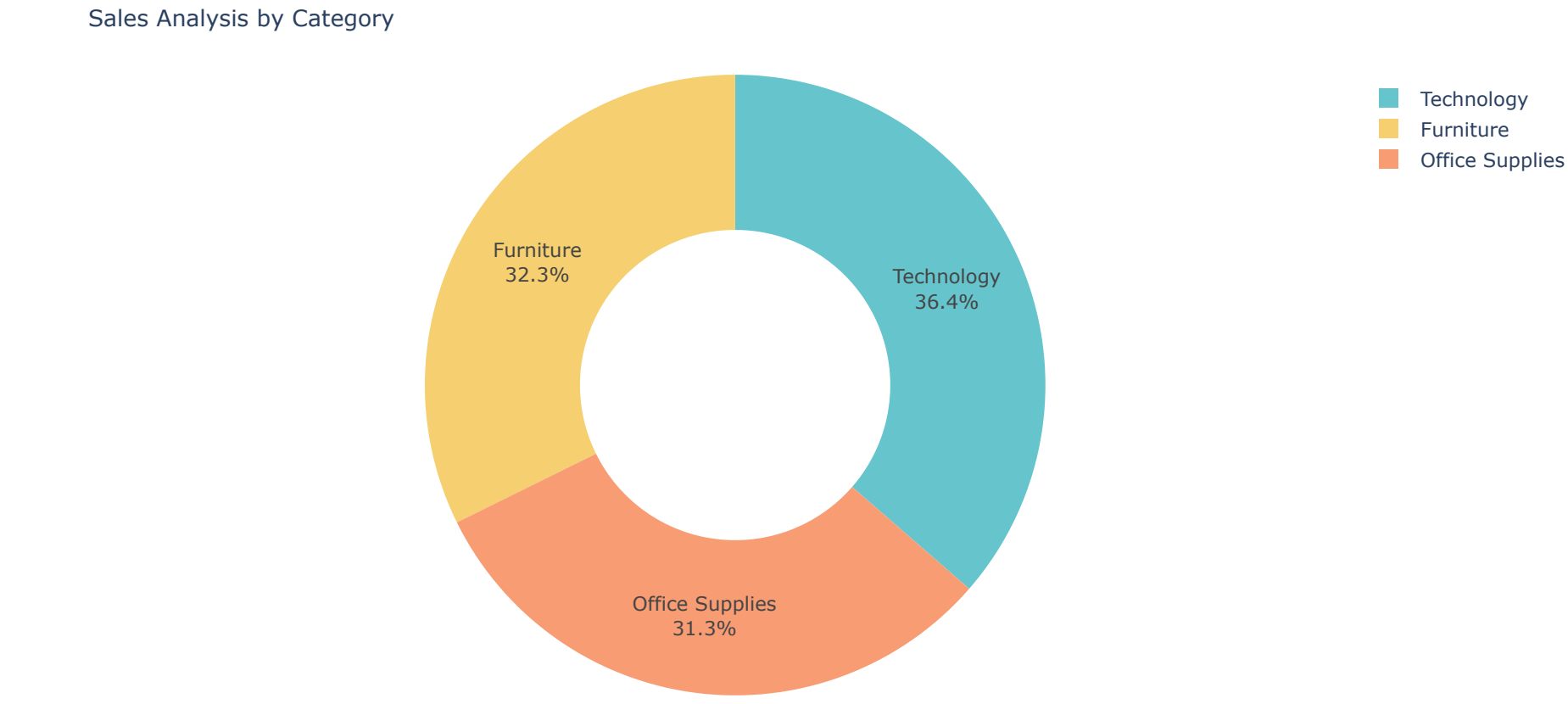
Analysis of sales by Category

In [8]:
```
print("Using Seaborn and Matplotlib library")
# Group by category and sum the sales
category_sales = df.groupby('Category')['Sales'].sum().reset_index()

# Sort the results by sales in descending order
category_sales = category_sales.sort_values(by='Sales', ascending=False)

# Set the color palette
sns.set_palette("pastel")

# Create a pie chart
plt.figure(figsize=(4, 4))
plt.pie(category_sales['Sales'], labels=category_sales['Category'], autopct='%1.1f%%')
plt.title('Sales Analysis by Category')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Show the plot
plt.show()

#*******************************************#

print("Using plotly.express library")

category_sales = df.groupby('Category')['Sales'].sum().reset_index()

fig = px.pie(category_sales,values='Sales',names='Category',hole=0.5,
             color_discrete_sequence=px.colors.qualitative.Pastel)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title_text='Sales Analysis by Category', title_font=dict(size=14))
fig.show()

```

```
Using Seaborn and Matplotlib library
```



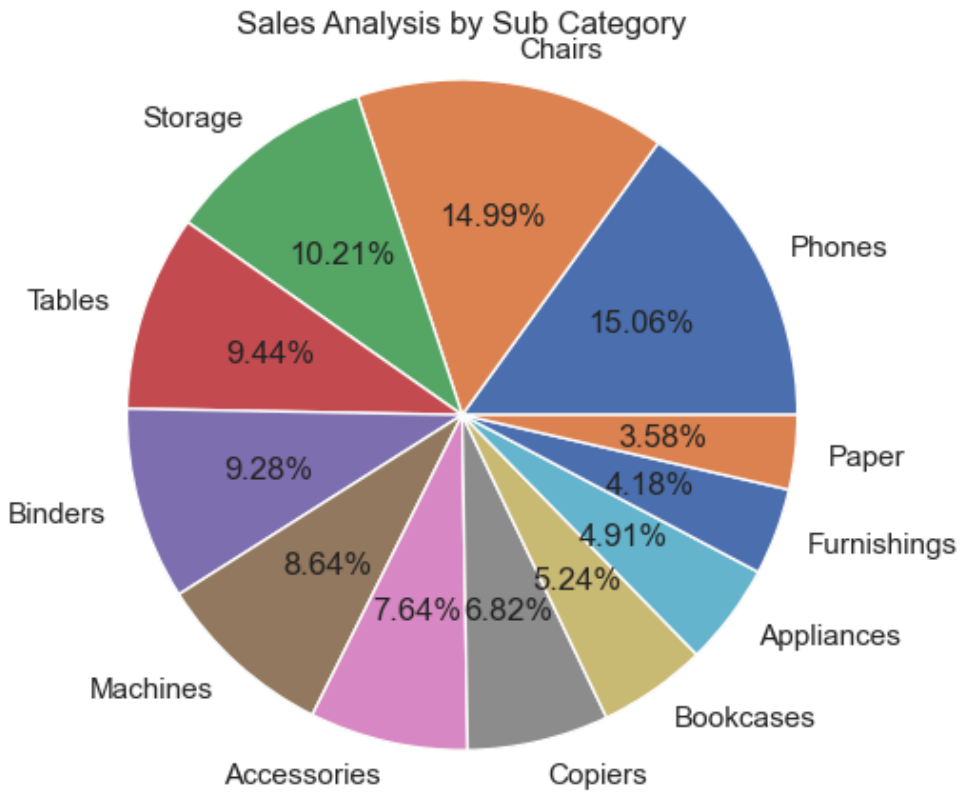

```
Using plotly.express library
```



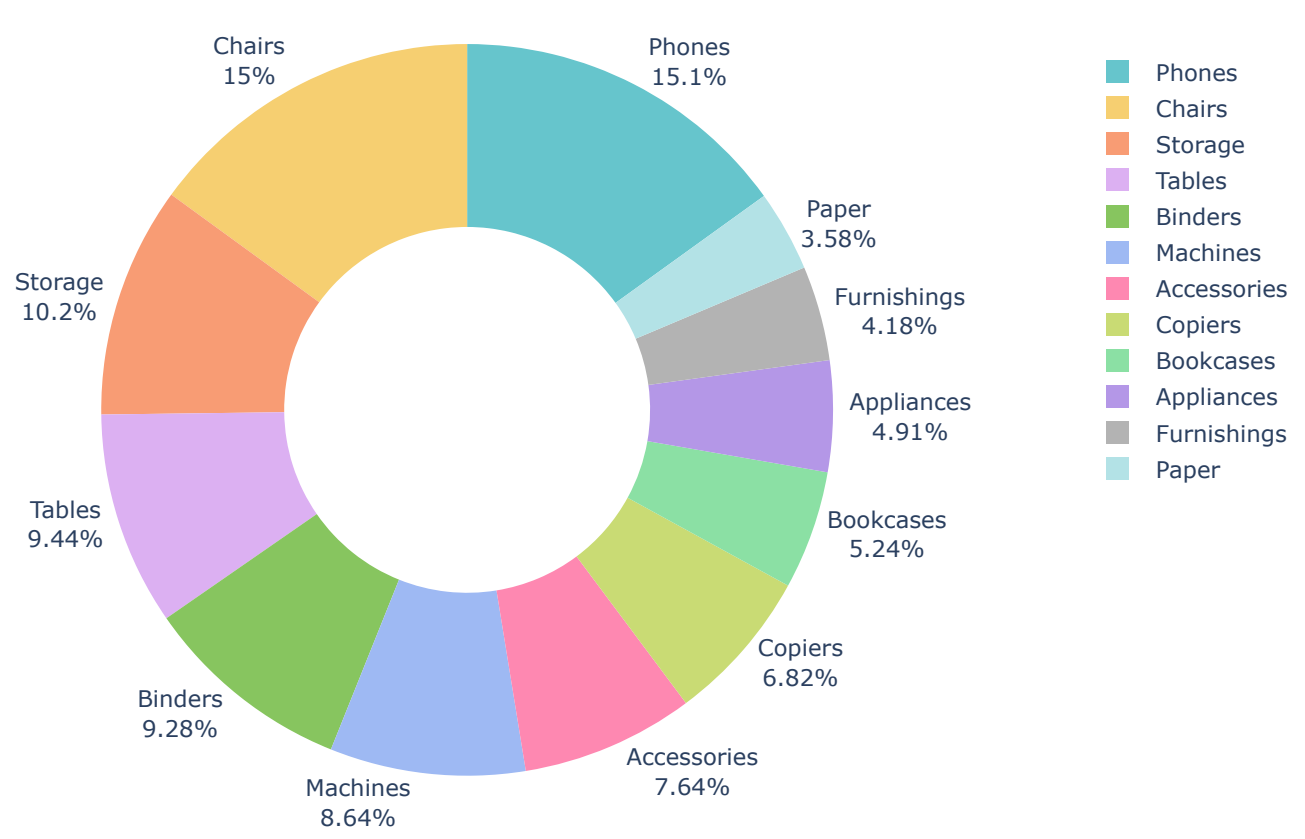
Analysis of sales by SubCategory

In [9]:
```
print("Using Seaborn and Matplotlib library")
# Group by Sub-Category and sum the sales
subcategory_sales = df.groupby('Sub-Category')['Sales'].sum().reset_index()

# Sort the results by sales in descending order
subcategory_sales = subcategory_sales.sort_values(by='Sales', ascending=False).head(12)

# Set the color palette
sns.set(style='whitegrid')

# Create a pie chart
plt.figure(figsize=(5, 5))
plt.pie(subcategory_sales['Sales'], labels=subcategory_sales['Sub-Category'], autopct='%0.2f%%')
plt.title('Sales Analysis by Sub Category')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Show the plot
plt.show()

#************************************************#
print("Using plotly.express library")

subcategory_sales = df.groupby('Sub-Category')['Sales'].sum().reset_index().nlargest(12,'Sales')

fig = px.pie(subcategory_sales,values='Sales',names='Sub-Category',hole=0.5,
             color_discrete_sequence=px.colors.qualitative.Pastel)
fig.update_traces(textposition='outside', textinfo='percent+label')
fig.update_layout(title_text='Sales Analysis by SubCategory', title_font=dict(size=18))
fig.show()

```

```
Using Seaborn and Matplotlib library
```




```
Using plotly.express library
```

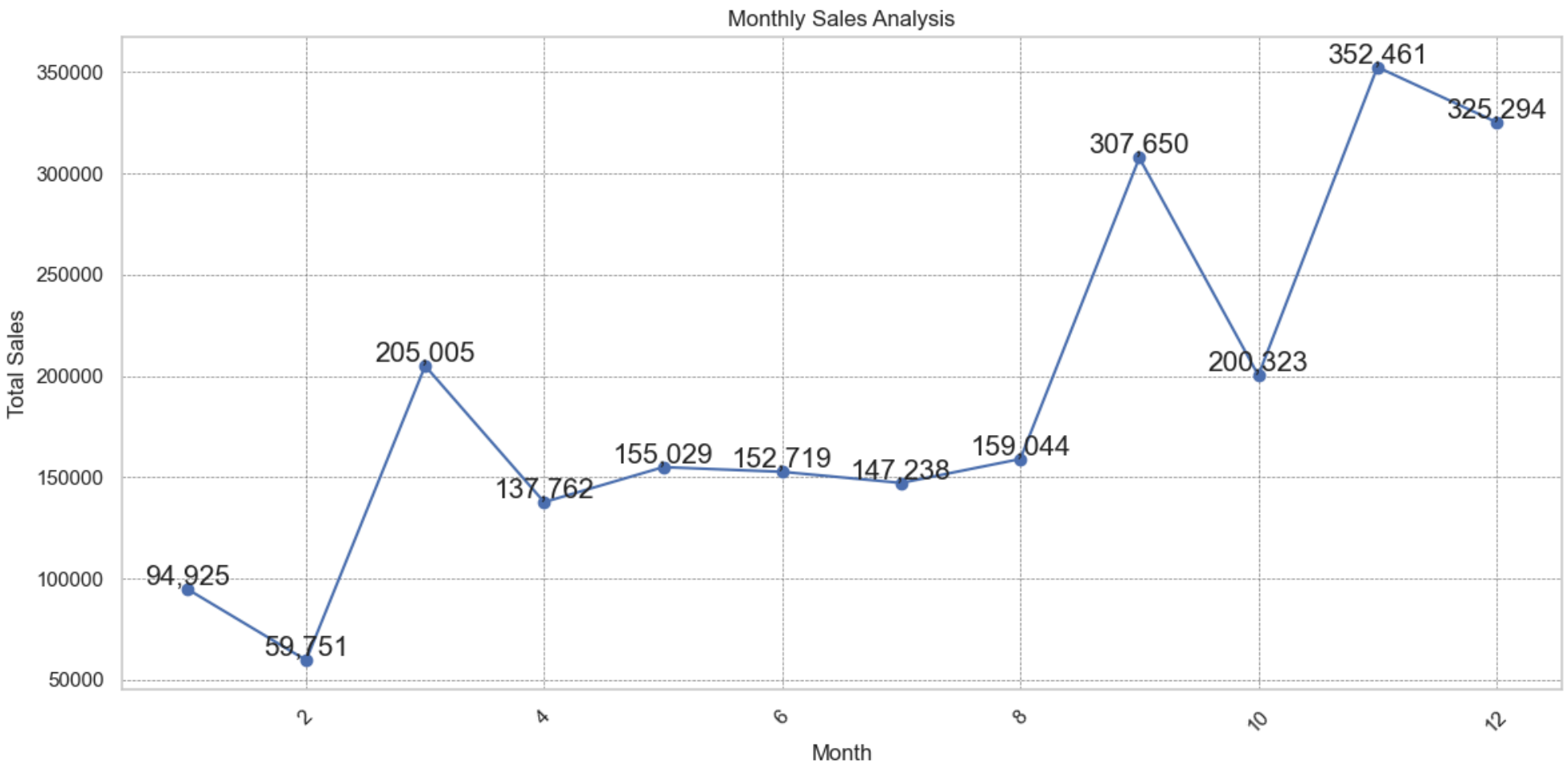


Monthly Sales Analysis

In [10]:

```
print("Using matplotlib.pyplot")

# Group by month and sum the sales
monthly_sales = df.groupby(['Order Month'])['Sales'].sum().reset_index()

# Plotting the monthly sales
plt.figure(figsize=(12, 6))
plt.plot(monthly_sales['Order Month'], monthly_sales['Sales'], marker='o')

# annotate each data point with its value
for i in range(len(monthly_sales)):
    plt.text(monthly_sales['Order Month'][i],monthly_sales['Sales'][i],
             f'{monthly_sales["Sales"][i]:,.0f}',fontsize=15, ha='center',va='bottom')

plt.title('Monthly Sales Analysis')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.grid(color='gray',linestyle='--',linewidth=0.5)
plt.tight_layout()
plt.show()


#*********************************************#
print("Using plotly.express")


monthly_sales=df.groupby('Order Month')['Sales'].sum().reset_index()
fig = px.line(monthly_sales, x = 'Order Month', y = 'Sales',
              title = 'Monthly Sales Anallysis',markers=True)

#add data labels to each point
for i,row in monthly_sales.iterrows():
    fig.add_annotation( x = row['Order Month'], y = row['Sales'],
                        text = f'{row["Sales"]:,.0f}',ax = 0, ay = -10)
fig.show()

```

```
Using matplotlib.pyplot
```



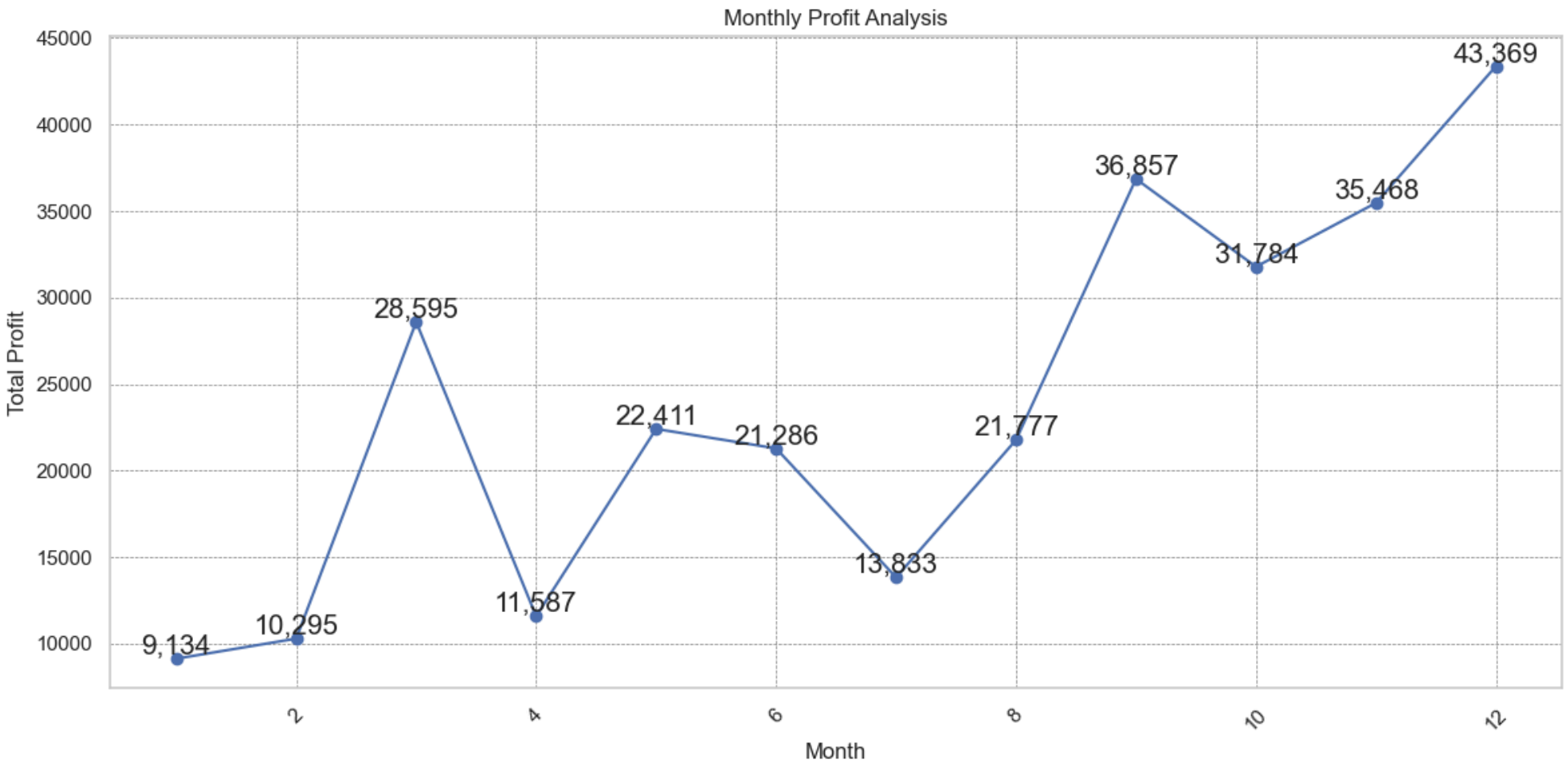

```
Using plotly.express
```

Monthly Profit Analysis

In [11]:

```
print("Using matplotlib.pyplot")

# Group by month and sum the Profit
monthly_profit = df.groupby(['Order Month'])['Profit'].sum().reset_index()

# Plotting the monthly Profit
plt.figure(figsize=(12, 6))
plt.plot(monthly_profit['Order Month'], monthly_profit['Profit'], marker='o')

# annotate each data point with its value
for i in range(len(monthly_profit)):
    plt.text(monthly_profit['Order Month'][i],monthly_profit['Profit'][i],
             f'{monthly_profit["Profit"][i]:,.0f}',fontsize=15, ha='center',va='bottom')


plt.title('Monthly Profit Analysis')
plt.xlabel('Month')
plt.ylabel('Total Profit')
plt.xticks(rotation=45)
plt.grid(color='gray',linestyle='--',linewidth=0.5)
plt.tight_layout()
plt.show()

#*******************************#
print("Using plotly.express")

monthly_profit=df.groupby('Order Month')['Profit'].sum().reset_index()
fig = px.line(monthly_profit, x = 'Order Month', y = 'Profit',
              title = 'Monthly Profit Anallysis',markers=True)

#add data labels to each point
for i,row in monthly_profit.iterrows():
    fig.add_annotation( x = row['Order Month'], y = row['Profit'],
                        text = f'{row["Profit"]:,.0f}',ax = 0, ay = -10)
fig.show()

```

Using matplotlib.pyplot

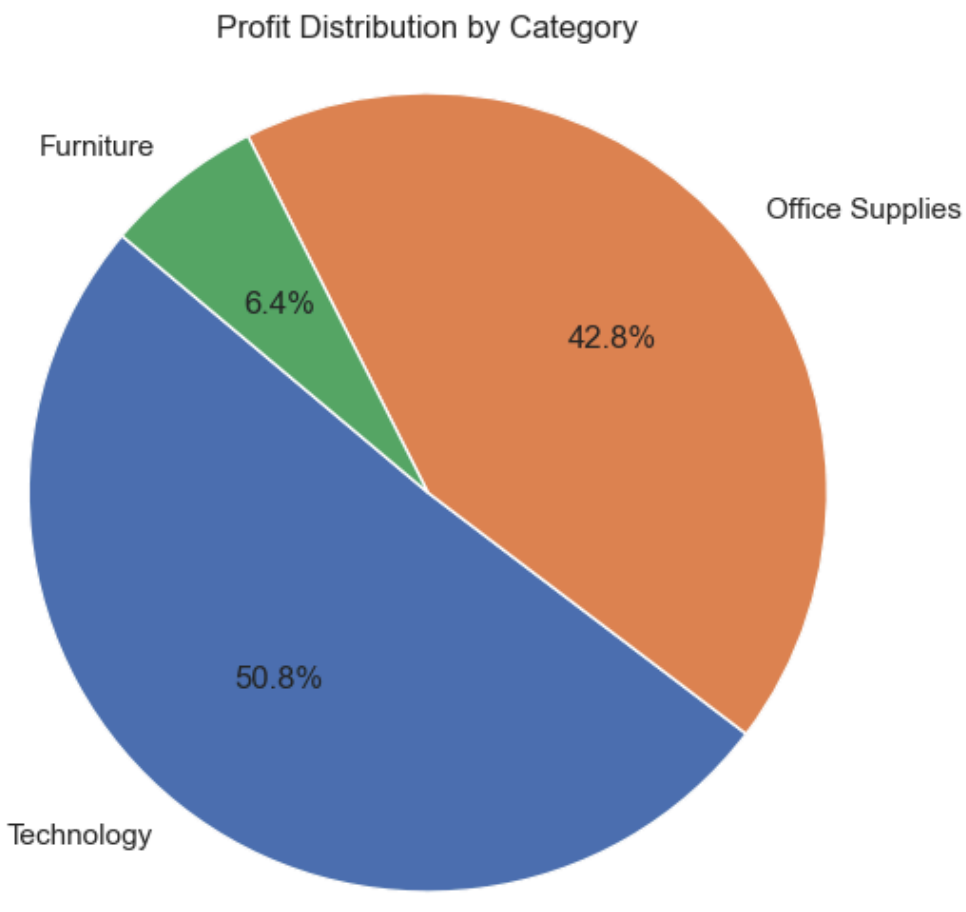


Using plotly.express

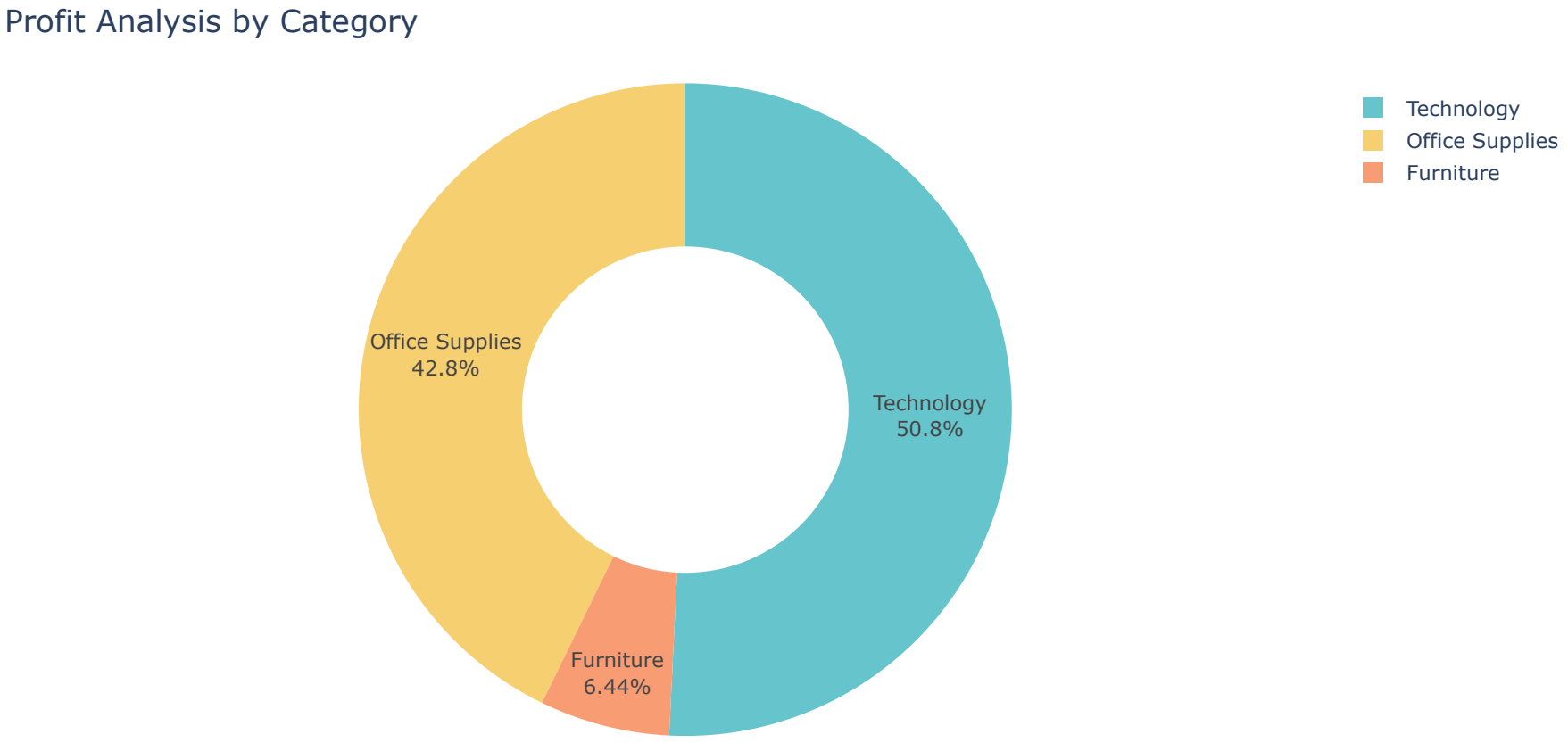
Profit Analysis by Category

In [12]:
```
print("using matplotlib library")

# Group by category and sum the profits
category_profit = df.groupby('Category')['Profit'].sum().reset_index()

# Sort the results by profit in descending order
category_profit = category_profit.sort_values(by='Profit', ascending=False)

# Create a pie chart
plt.figure(figsize=(7, 6))
plt.pie(category_profit['Profit'], labels=category_profit['Category'], autopct='%1.1f%%', startangle=140)
plt.title('Profit Distribution by Category')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Show the plot
plt.show()



#*******************************************#
print("using plotly.express library")

# Group by category and sum the profits
category_profit = df.groupby('Category')['Profit'].sum().reset_index()

# Create a pie chart
fig= px.pie(category_profit,values='Profit', names='Category',hole=0.5,
           color_discrete_sequence=px.colors.qualitative.Pastel)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title_text='Profit Analysis by Category', title_font=dict(size=18))

# Show the plot
fig.show()

```

```
using matplotlib library
```




```
using plotly.express library
```

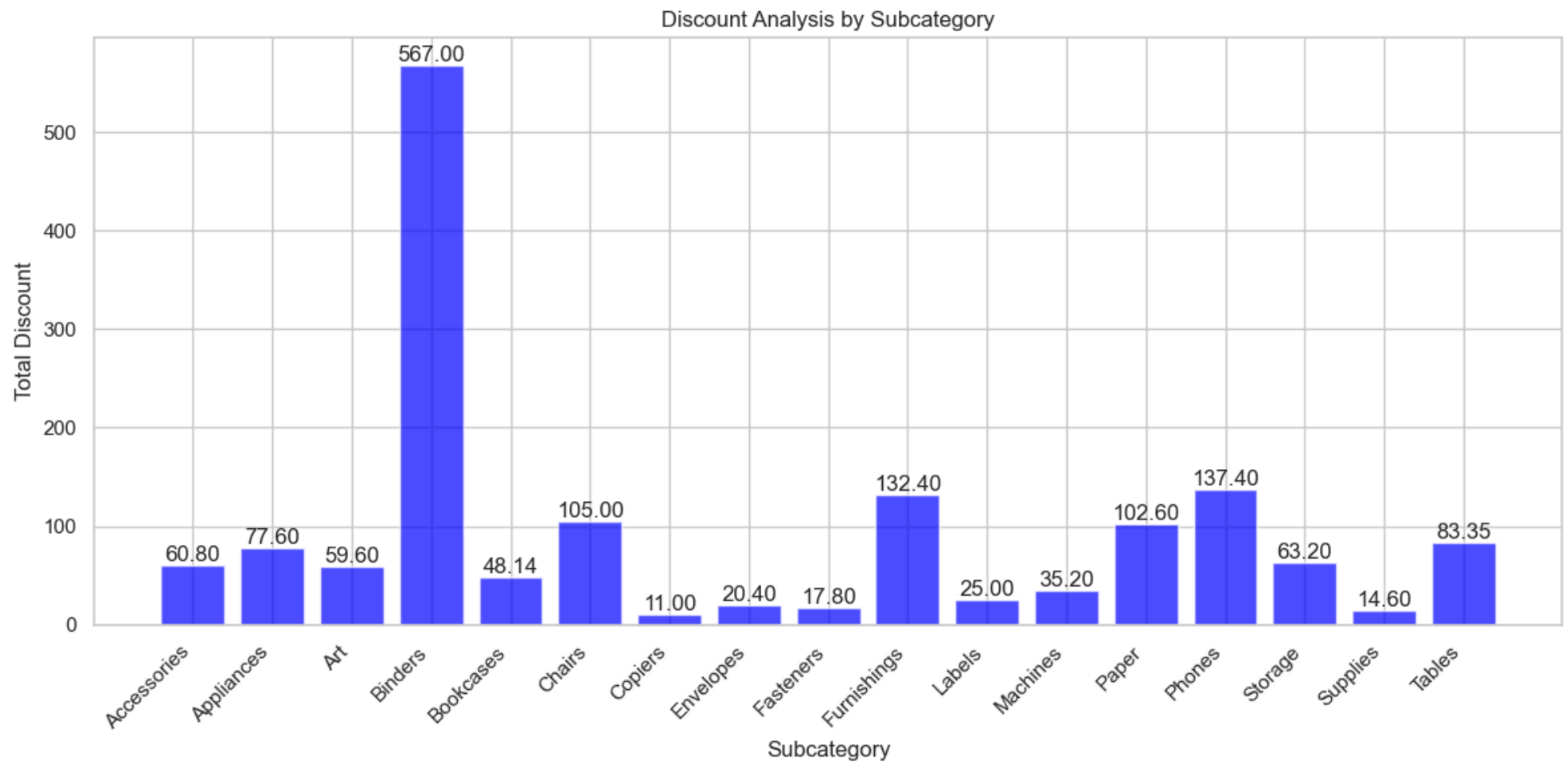


Discount Analysis by Subcategory

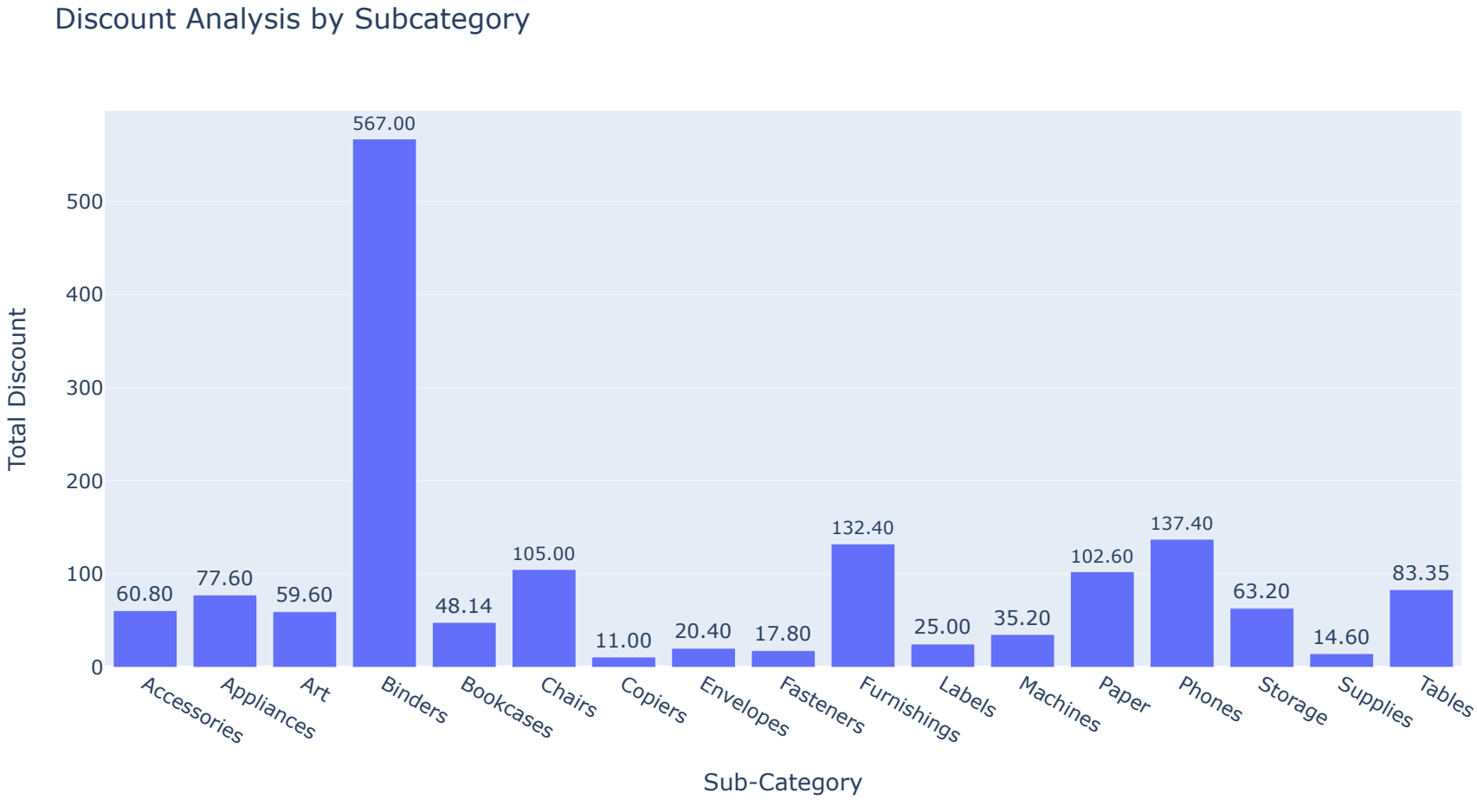
In [13]:

```
print("Using Matplotlib library")

# Group by Subcategory and sum the Discounts
subcategory_analysis = df.groupby('Sub-Category')['Discount'].sum().reset_index()

# Set the figure size
plt.figure(figsize=(12, 6))

# Create a bar chart
bars = plt.bar(subcategory_analysis['Sub-Category'], subcategory_analysis['Discount'], color='blue', alpha=0.7)

# Add titles and labels
plt.title('Discount Analysis by Subcategory')
plt.xlabel('Subcategory')
plt.ylabel('Total Discount')

# Annotate the bars with their values without a loop
plt.bar_label(bars, label_type='edge', fmt='%.2f')

# Rotate x-ticks for better readability
plt.xticks(rotation=45, ha='right')

# Show the plot
plt.tight_layout()
plt.show()


#****************************************#
print("Using Plotly.Express library")

# Group by Subcategory and sum the Discounts
subcategory_analysis = df.groupby('Sub-Category')['Discount'].sum().reset_index()

# Create a bar chart using Plotly Express
fig = px.bar(subcategory_analysis, x='Sub-Category', y='Discount',
             title='Discount Analysis by Subcategory',
             labels={'Discount': 'Total Discount'})

# Add data labels to each bar
fig.update_traces(texttemplate='%{y:,.2f}', textposition='outside')

# Show the plot
fig.show()

```

```
Using Matplotlib library
```




```
Using Plotly.Express library
```

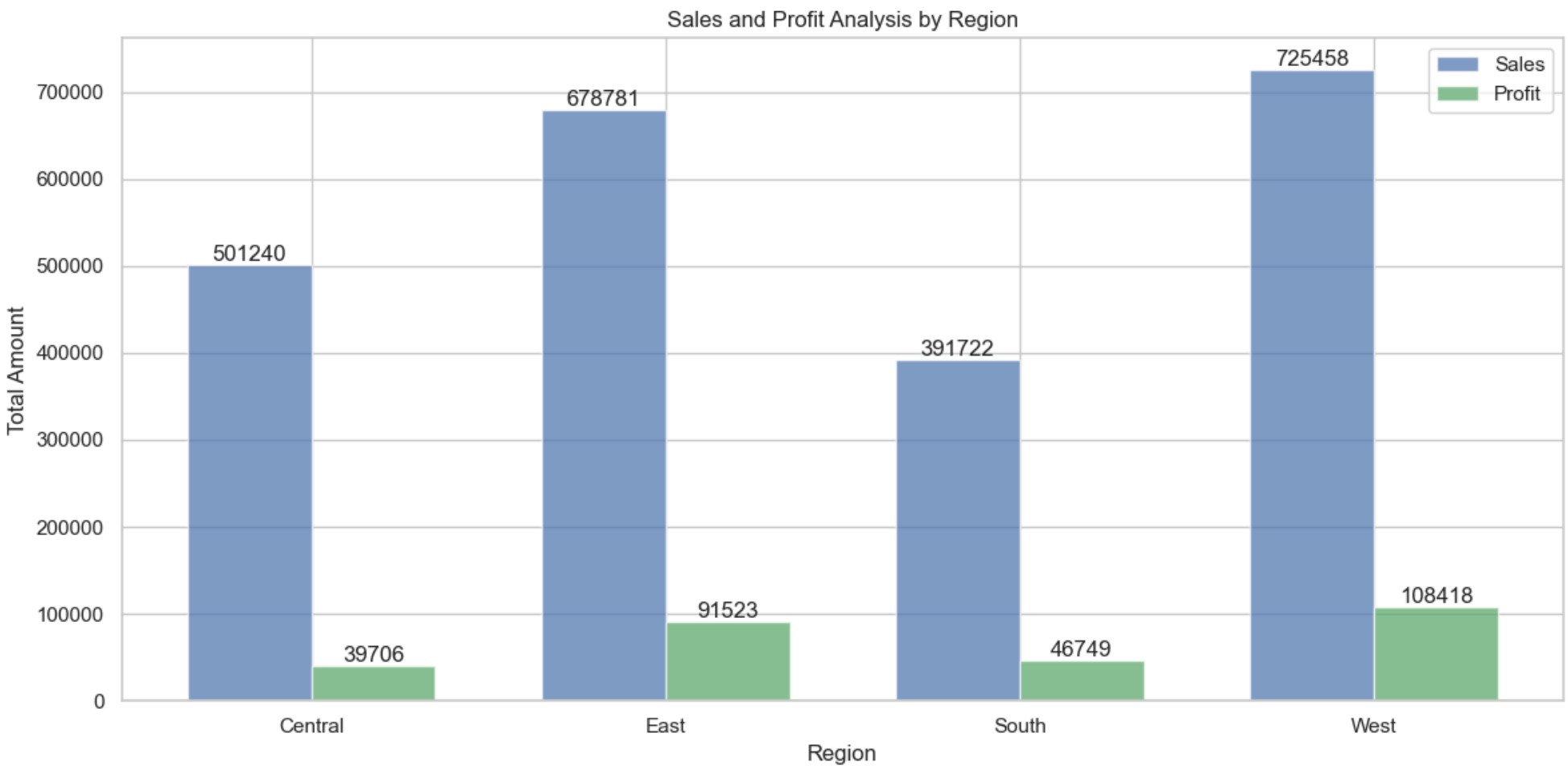


Sales and Profit Analysis by Region

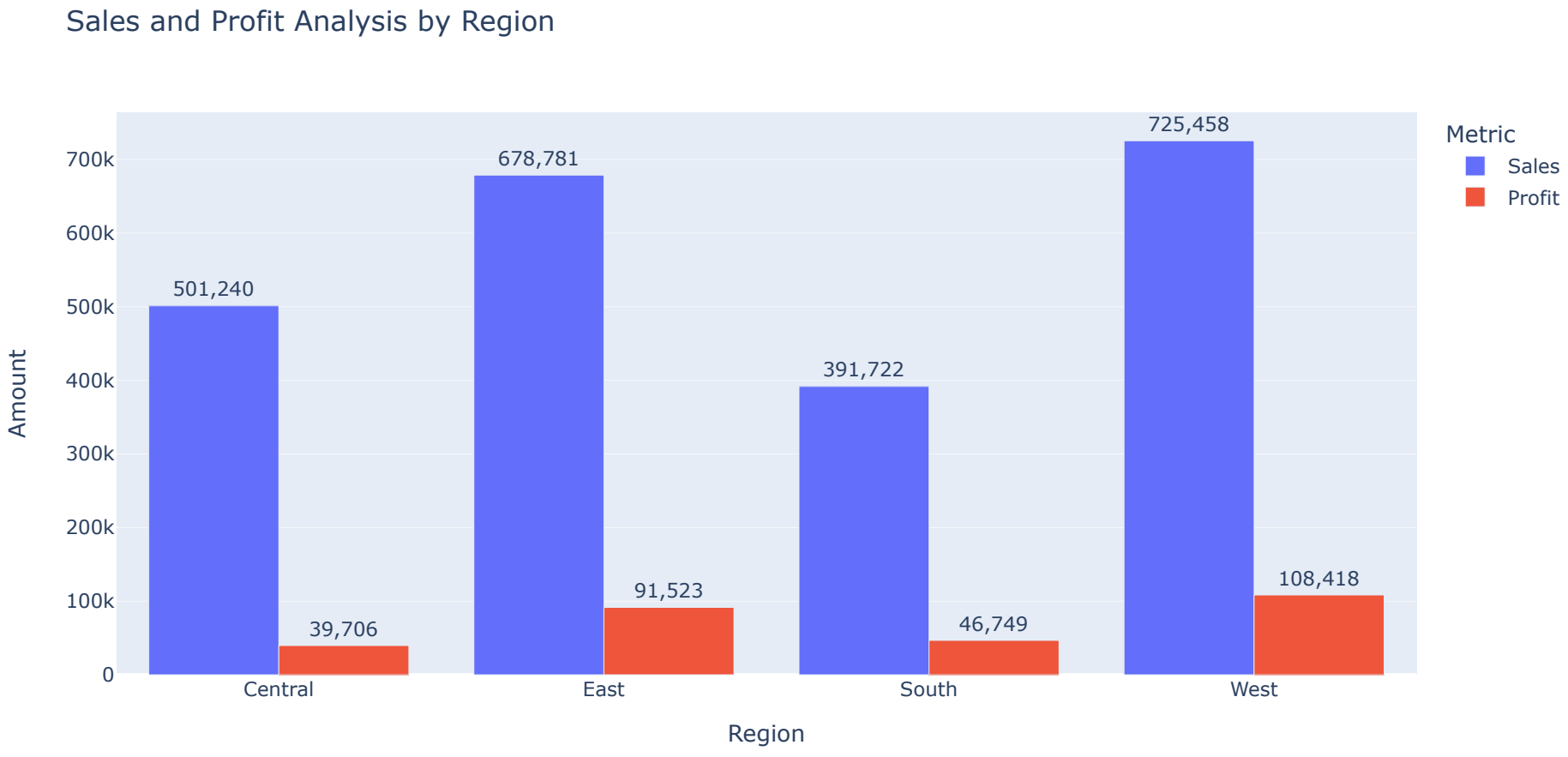
In [14]:

```
print("Using Matplotlib library")

# Group by Region and sum the Sales and Profit
region_analysis = df.groupby('Region')[['Sales', 'Profit']].sum().reset_index()

# Set the figure size
plt.figure(figsize=(12, 6))

# Create bar plots for Sales and Profit
bar_width = 0.35
x = range(len(region_analysis))

# Create bars for Sales
sales_bars = plt.bar(x, region_analysis['Sales'], width=bar_width, label='Sales',
                     color='b', alpha=0.7)

# Create bars for Profit
profit_bars = plt.bar([p + bar_width for p in x], region_analysis['Profit'],
                      width=bar_width, label='Profit', color='g', alpha=0.7)

# Set the x-ticks and labels
plt.xticks([p + bar_width / 2 for p in x], region_analysis['Region'])

# Add titles and labels
plt.title('Sales and Profit Analysis by Region')
plt.xlabel('Region')
plt.ylabel('Total Amount')
plt.legend()

#Annotate the bars with values
plt.bar_label(sales_bars,label_type='edge',fmt='%.0f')
plt.bar_label(profit_bars,label_type='edge',fmt='%.0f')
# Show the plot
plt.tight_layout()
plt.show()

#************************************************#
print("Using Plotly.Express library")

# Group by Region and sum the Sales and Profit
region_analysis = df.groupby('Region')[['Sales', 'Profit']].sum().reset_index()

# Melt the DataFrame for Plotly
region_melted = region_analysis.melt(id_vars='Region', value_vars=['Sales', 'Profit'],
                                     var_name='Metric', value_name='Amount')

# Create a bar chart using Plotly Express
fig = px.bar(region_melted, x='Region', y='Amount', color='Metric',
             title='Sales and Profit Analysis by Region',
             barmode='group')

#Add data labels to each bar
fig.update_traces(texttemplate='%{y:,.0f}',textposition='outside')
# Show the plot
fig.show()

```

```
Using Matplotlib library
```



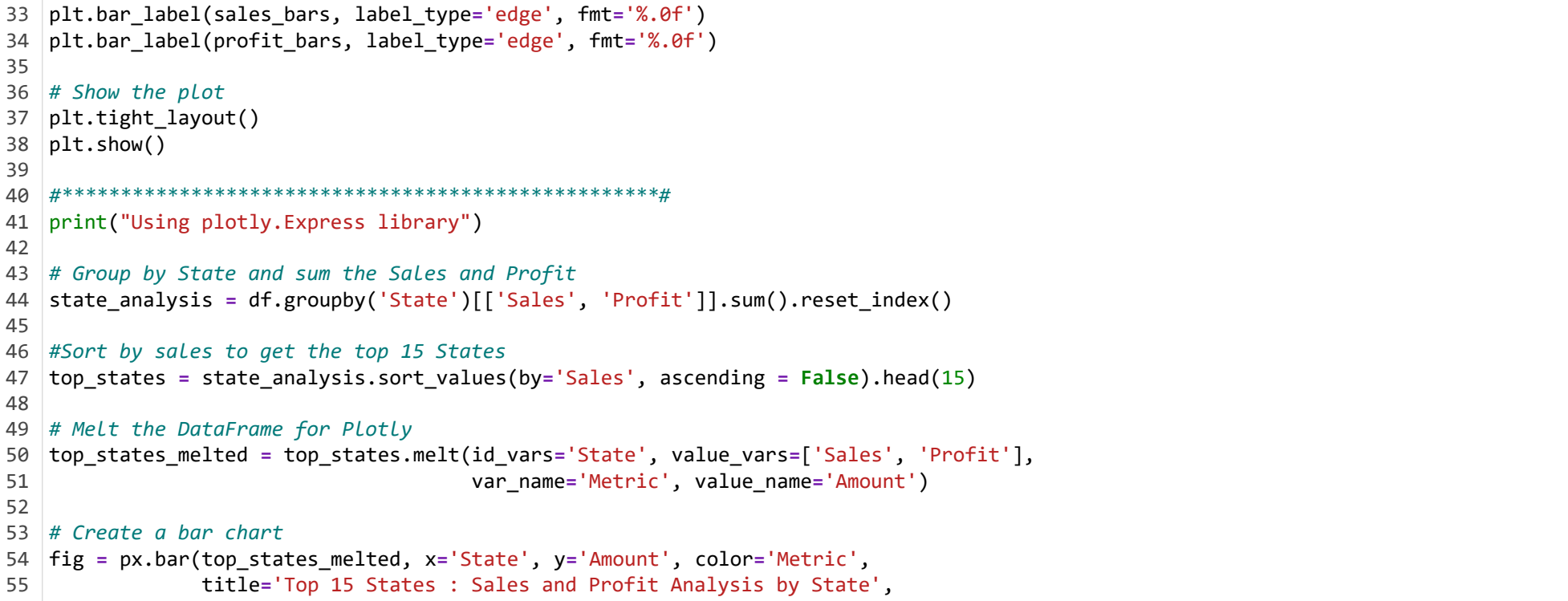

```
Using Plotly.Express library
```



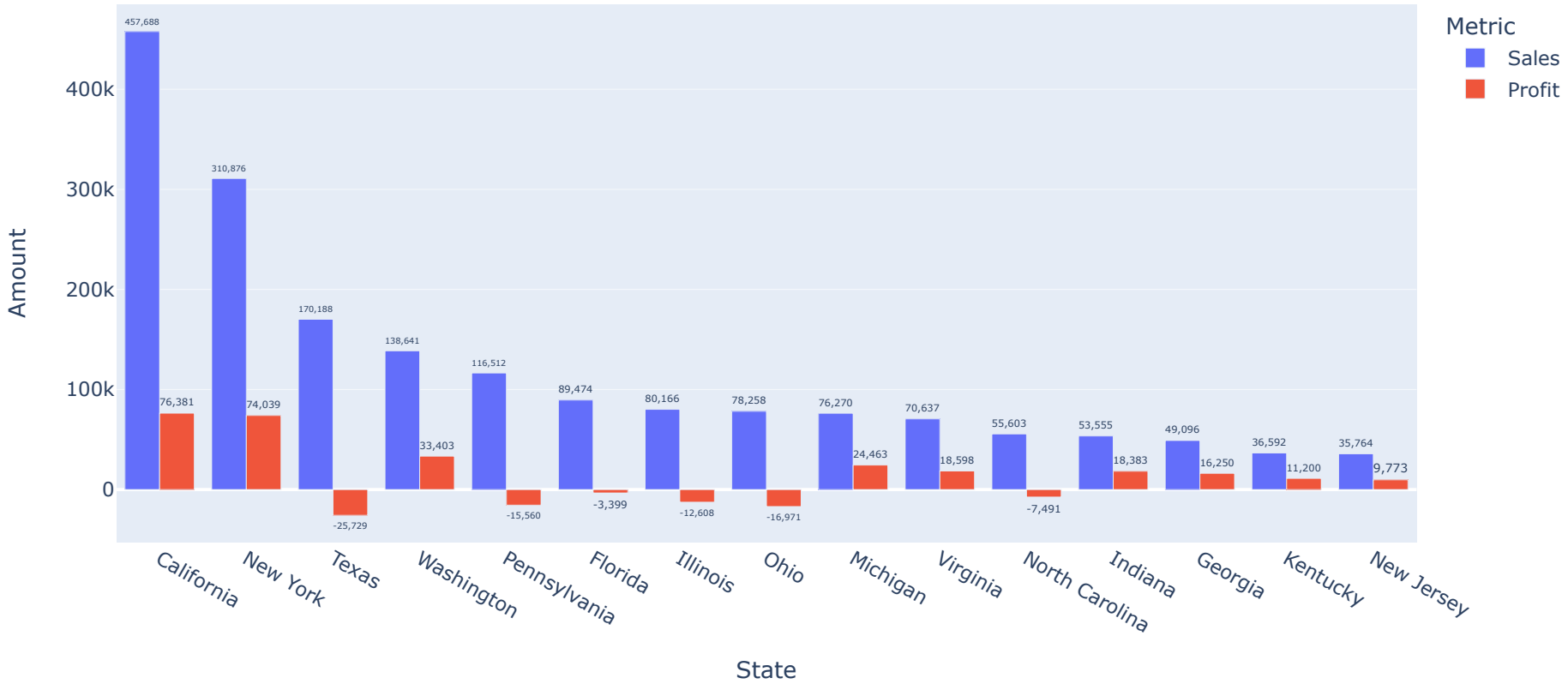
Sales and Profit Analysis by Statewise

In [15]:
```
print("Using Matplotlib library")

# Group by State and sum the Sales and Profit
state_analysis = df.groupby('State')[['Sales', 'Profit']].sum().reset_index()

#Sort by sales to get the top 15 States
top_states = state_analysis.sort_values(by='Sales', ascending = False).head(15)

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

# Set bar width and positions
bar_width = 0.35
x = range(len(top_states))

# Create bars for Sales
sales_bars = plt.bar(x, top_states['Sales'], width=bar_width, label='Sales', color='b', alpha=0.7)

# Create bars for Profit
profit_bars = plt.bar([p + bar_width for p in x], top_states['Profit'], width=bar_width, label='Profit',
                      color='red', alpha=0.7)

# Set the x-ticks and labels
plt.xticks([p + bar_width / 2 for p in x], top_states['State'])

# Add titles and labels
plt.title('Top 15 States : Sales and Profit Analysis by State')
plt.xlabel('State')
plt.ylabel('Total Amount')
plt.legend()

# Annotate the bars with values
plt.bar_label(sales_bars, label_type='edge', fmt='%.0f')
plt.bar_label(profit_bars, label_type='edge', fmt='%.0f')

# Show the plot
plt.tight_layout()
plt.show()

#************************************************#
print("Using plotly.Express library")

# Group by State and sum the Sales and Profit
state_analysis = df.groupby('State')[['Sales', 'Profit']].sum().reset_index()

#Sort by sales to get the top 15 States
top_states = state_analysis.sort_values(by='Sales', ascending = False).head(15)

# Melt the DataFrame for Plotly
top_states_melted = top_states.melt(id_vars='State', value_vars=['Sales', 'Profit'],
                                    var_name='Metric', value_name='Amount')

# Create a bar chart
fig = px.bar(top_states_melted, x='State', y='Amount', color='Metric',
             title='Top 15 States : Sales and Profit Analysis by State',
             barmode='group')

# Add data labels to each bar
fig.update_traces(texttemplate='%{y:,.0f}', textposition='outside')

# Show the plot
fig.show()

```

```
Using Matplotlib library
```




```
Using plotly.Express library
```



summarizing key insights from a sales dataset

1. Category Sales Contribution: • Technology has the highest sales at 36.4%.

• The highest sales contribution in the subcategory is from Phones, accounting for 15.06% of sales.

2. Monthly Sales Performance:

• The months of November and December show the highest sales contributions.

3. Profit Insights:

• The highest profit occurs from September to December.

• The peak profit of ₹43,369 is recorded in December.

• Technology leads in profit contribution at 50.8%, followed by Office Supplies at 42.8%, and Furniture at 6.4%.

4. Discount Contributions:

• Binders have the maximum discount contribution at ₹567.

5. Regional Sales and Profit:

• The West Region contributes the most to sales and profit, totaling ₹725,458, followed by the East, Central, and South Regions (least at ₹391,722).

6. State Contributions:

• Among the top 15 states, California tops with sales of ₹457,688, followed closely by New York and Texas.

• Profit Contribution are as follows: California: ₹76,381 New York: ₹74,039 Texas shows a loss of -₹25,729.

Conclusion

This analysis highlights significant trends in sales and profit across various categories, months, regions, and states. Technology consistently outperforms in both sales and profit, while the West Region dominates overall contributions. Understanding these dynamics can guide strategic decisions in inventory, marketing, and regional focus for improved performance.

In [ ]:
```
1
```