E-commerce_data_analysis

February 18, 2025

[2]:
```
# Importing Libraries
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as pc
pio.templates.default = "plotly_dark"

import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
# Let's read the data
df = pd.read_csv('Superstore.csv' , encoding= 'latin-1')
# max column width to display
pd.set_option('display.max_columns', None)

df.head(3)
```

[3]:

	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
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.96	2
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,…	731.94	3
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b…	14.62	2

```
   Discount    Profit
0       0.0   41.9136
1       0.0  219.5820
2       0.0    6.8714
```

[4]:
```
# Let's check the data info
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
```

[5]:
```
# # Let's check the data types of the columns
df.dtypes
```

[5]:
```
Row ID            int64
Order ID         object
Order Date       object
Ship Date        object
Ship Mode        object
Customer ID      object
Customer Name    object
Segment          object
```

```
Country           object
City              object
State             object
Postal Code        int64
Region            object
Product ID        object
Category          object
Sub-Category      object
Product Name      object
Sales            float64
Quantity           int64
Discount         float64
Profit           float64
dtype: object
```

[6]:
```
# There are some columns which are not in the correct data type. Let's convert␣
 ↪them to the correct data type.
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

[7]:
```
# let's check it again
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   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 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: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

[8]:
```
df.head(2)
```

[8]:
```
   Row ID        Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0       1  CA-2016-152156 2016-11-08 2016-11-11  Second Class    CG-12520
1       2  CA-2016-152156 2016-11-08 2016-11-11  Second Class    CG-12520

  Customer Name   Segment        Country       City     State  Postal Code  \
0   Claire Gute  Consumer  United States  Henderson  Kentucky        42420
1   Claire Gute  Consumer  United States  Henderson  Kentucky        42420

  Region       Product ID   Category Sub-Category  \
0  South  FUR-BO-10001798  Furniture    Bookcases
1  South  FUR-CH-10000454  Furniture       Chairs

                                        Product Name   Sales  Quantity  \
0                  Bush Somerset Collection Bookcase  261.96         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,…  731.94         3

   Discount    Profit
0       0.0   41.9136
1       0.0  219.5820
```

[9]:
```
# let's check the shape of the data
print('Rows:', df.shape[0] , 'Columns:', df.shape[1])
```

```
Rows: 9994 Columns: 21
```

[10]:
```
# Let's check the discriptive statistics of the data
df.describe()
```

[10]:
```
            Row ID                      Order Date  \
count  9994.000000                            9994
mean   4997.500000  2016-04-30 00:07:12.259355648
min       1.000000            2014-01-03 00:00:00
25%    2499.250000            2015-05-23 00:00:00
50%    4997.500000            2016-06-26 00:00:00
75%    7495.750000            2017-05-14 00:00:00
max    9994.000000            2017-12-30 00:00:00
std    2885.163629                            NaN

                           Ship Date   Postal Code         Sales     Quantity  \
count                           9994   9994.000000   9994.000000  9994.000000
mean   2016-05-03 23:06:58.571142912  55190.379428    229.858001     3.789574
min              2014-01-07 00:00:00   1040.000000      0.444000     1.000000
```

```
25%                  2015-05-27 00:00:00  23223.000000     17.280000     2.000000
50%                  2016-06-29 00:00:00  56430.500000     54.490000     3.000000
75%                  2017-05-18 00:00:00  90008.000000    209.940000     5.000000
max                  2018-01-05 00:00:00  99301.000000  22638.480000    14.000000
std                                  NaN  32063.693350    623.245101     2.225110

          Discount       Profit
count  9994.000000  9994.000000
mean      0.156203    28.656896
min       0.000000 -6599.978000
25%       0.000000     1.728750
50%       0.200000     8.666500
75%       0.200000    29.364000
max       0.800000  8399.976000
std       0.206452   234.260108
```

[11]:
```
# Let's check the missing values in the data
df.isnull().sum()
```

[11]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

- no-null values

[12]:
```
# let's create a year & month new columns from the Order Date column
df['Order_Month'] = df['Order Date'].dt.month
df['Order_Year'] = df['Order Date'].dt.year
# let's add one more column for the Order Day
```

```
df['Order_Day'] = df['Order Date'].dt.day_name()
```

[13]:
```
# let's check the data
df.head(2)
```

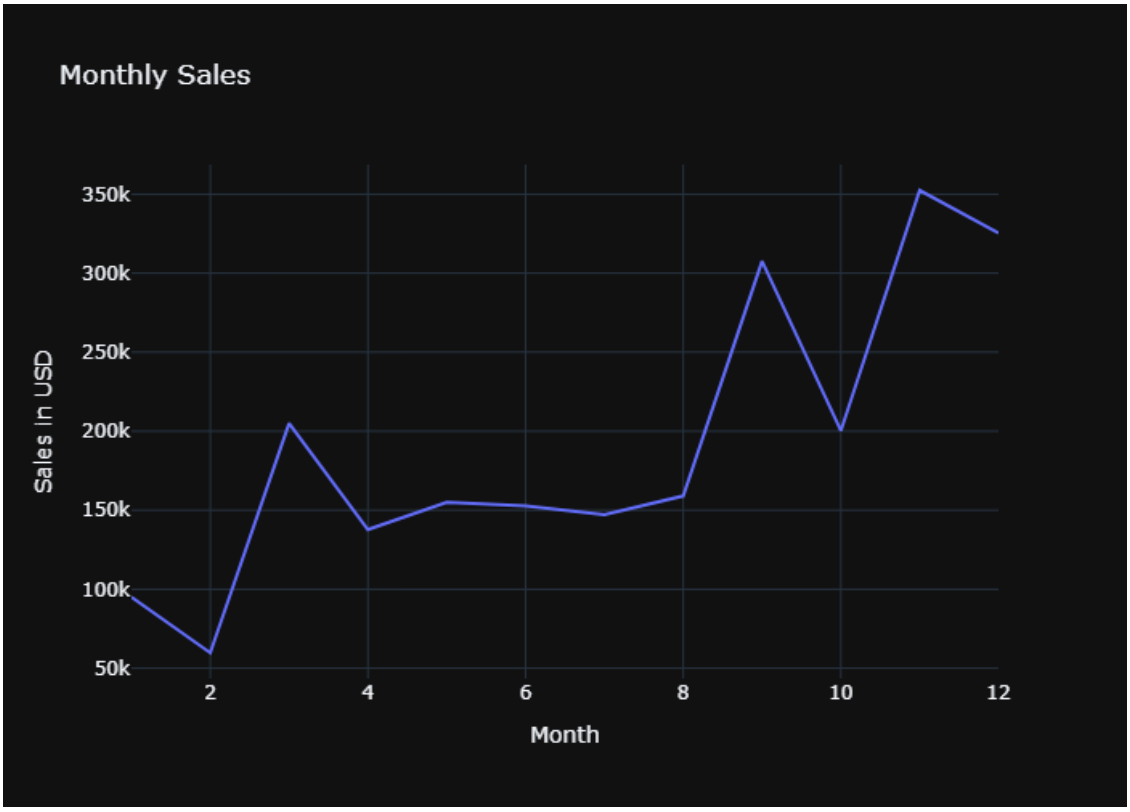
[13]:
```
   Row ID        Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0       1  CA-2016-152156 2016-11-08 2016-11-11  Second Class    CG-12520
1       2  CA-2016-152156 2016-11-08 2016-11-11  Second Class    CG-12520

  Customer Name   Segment        Country       City     State  Postal Code  \
0   Claire Gute  Consumer  United States  Henderson  Kentucky        42420
1   Claire Gute  Consumer  United States  Henderson  Kentucky        42420

  Region       Product ID   Category Sub-Category  \
0  South  FUR-BO-10001798  Furniture    Bookcases
1  South  FUR-CH-10000454  Furniture       Chairs

                                        Product Name   Sales  Quantity  \
0                  Bush Somerset Collection Bookcase  261.96         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,…  731.94         3

   Discount    Profit  Order_Month  Order_Year Order_Day
0       0.0   41.9136           11        2016   Tuesday
1       0.0  219.5820           11        2016   Tuesday
```

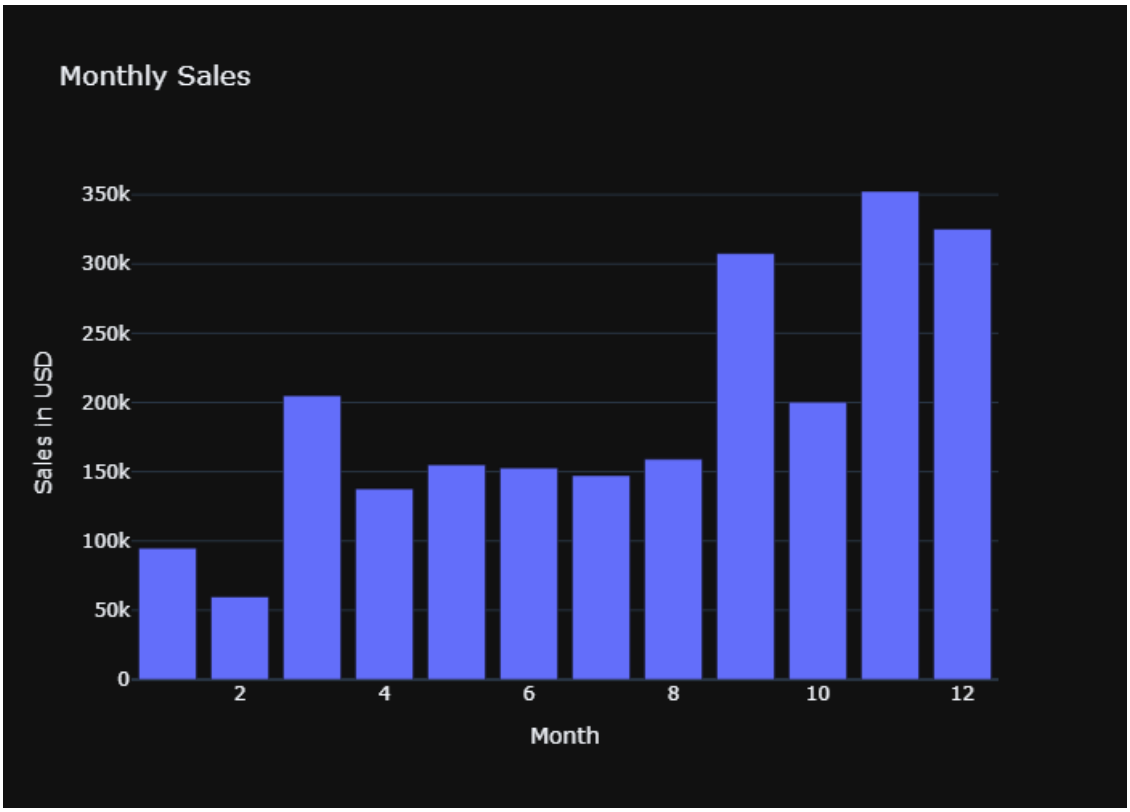
Let's analyse the data

[35]:
```
# let's check the monthly sales
monthly_sales = df.groupby('Order_Month')['Sales'].sum().reset_index()
print(monthly_sales)

# let's create a figure of monthly sales
fig1 = px.line(monthly_sales,
            x='Order_Month',
            y='Sales',
            title='Monthly Sales',
            labels={'Sales':'Sales in USD', 'Order_Month':'Month'})
fig1.show()
fig2 = px.bar(monthly_sales,
            x='Order_Month',
            y='Sales',
            title='Monthly Sales',
            labels={'Sales':'Sales in USD', 'Order_Month':'Month'})
fig2.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig1, file='monthly_sales_line.png')
```

```
# pio.write_image(fig2, file='monthly_sales_bar.png')
from IPython.display import Image , display
display(Image(filename='monthly_sales_line.png'))
display(Image(filename='monthly_sales_bar.png'))
```

	Order_Month	Sales
0	1	94924.8356
1	2	59751.2514
2	3	205005.4888
3	4	137762.1286
4	5	155028.8117
5	6	152718.6793
6	7	147238.0970
7	8	159044.0630
8	9	307649.9457
9	10	200322.9847
10	11	352461.0710
11	12	325293.5035



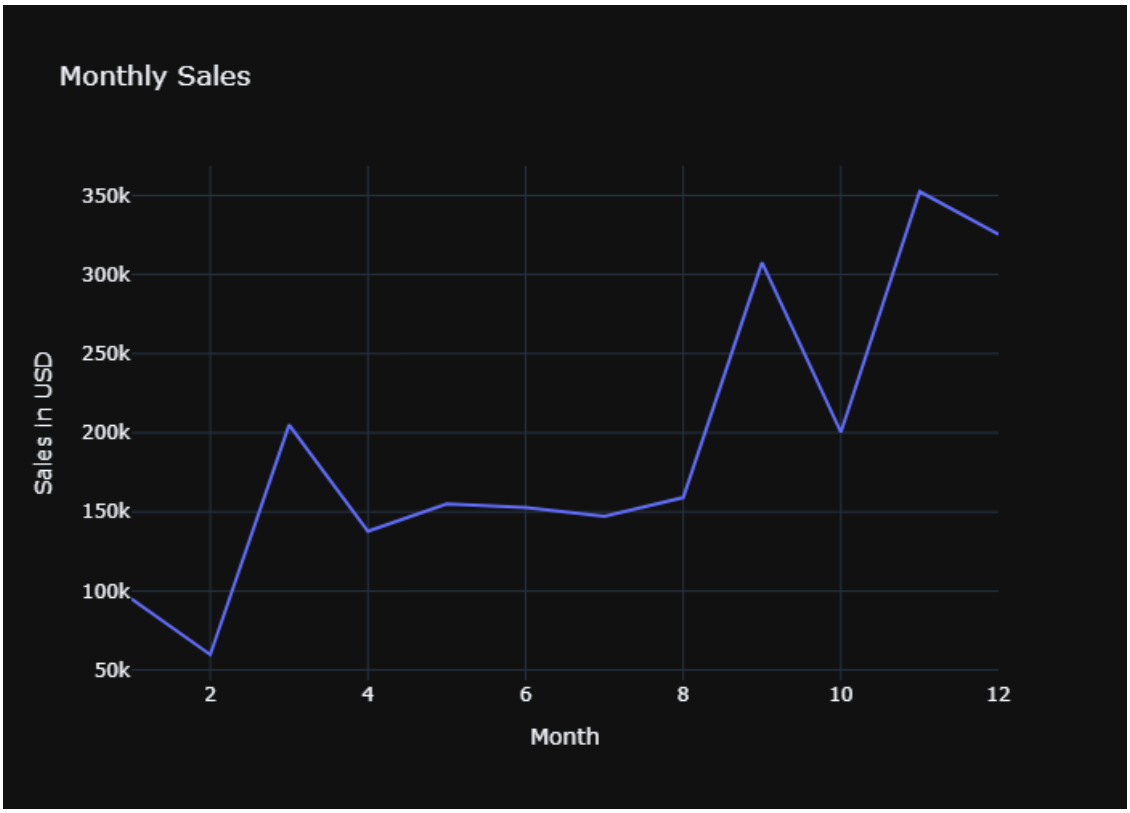


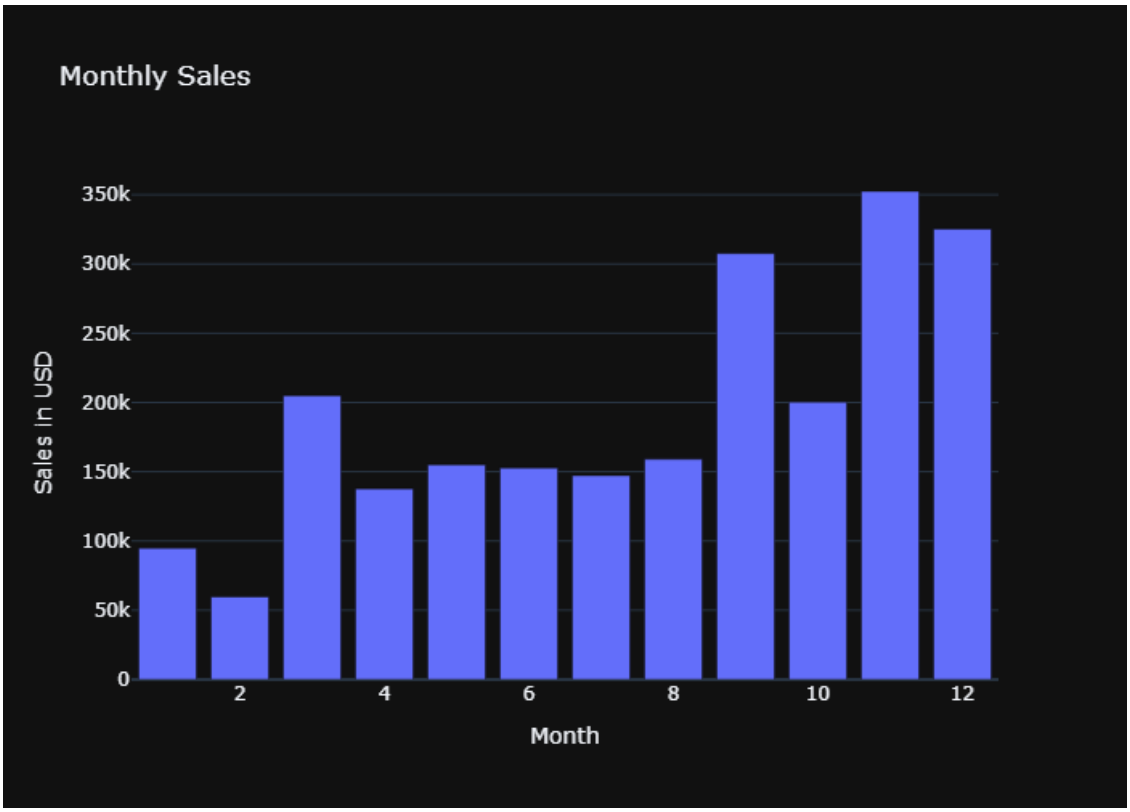
[37]:
```
# let's check the monthly sales of every year
monthly_sales_year = df.groupby(['Order_Year','Order_Month'])['Sales'].sum().
  ↪reset_index()
print(monthly_sales_year)
# let's create a figure of monthly sales of every year
fig = px.line(monthly_sales_year,
              x='Order_Month',
              y='Sales',
              title='Monthly Sales of Every Year',
              labels={'Sales':'Sales in USD', 'Order_Month':'Month'},
              color='Order_Year')
fig.show()
fig = px.bar(monthly_sales_year,
              x='Order_Month',
              y='Sales',
              title='Monthly Sales of Every Year',
              labels={'Sales':'Sales in USD', 'Order_Month':'Month'},
              color='Order_Year')
fig.show()
```

```
# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig1, file='monthly_sales_year_line.png')
# pio.write_image(fig2, file='monthly_sales_year_bar.png')
from IPython.display import Image , display
display(Image(filename='monthly_sales_year_line.png'))
display(Image(filename='monthly_sales_year_bar.png'))
```

	Order_Year	Order_Month	Sales
0	2014	1	14236.8950
1	2014	2	4519.8920
2	2014	3	55691.0090
3	2014	4	28295.3450
4	2014	5	23648.2870
5	2014	6	34595.1276
6	2014	7	33946.3930
7	2014	8	27909.4685
8	2014	9	81777.3508
9	2014	10	31453.3930
10	2014	11	78628.7167
11	2014	12	69545.6205
12	2015	1	18174.0756
13	2015	2	11951.4110
14	2015	3	38726.2520
15	2015	4	34195.2085
16	2015	5	30131.6865
17	2015	6	24797.2920
18	2015	7	28765.3250
19	2015	8	36898.3322
20	2015	9	64595.9180
21	2015	10	31404.9235
22	2015	11	75972.5635
23	2015	12	74919.5212
24	2016	1	18542.4910
25	2016	2	22978.8150
26	2016	3	51715.8750
27	2016	4	38750.0390
28	2016	5	56987.7280
29	2016	6	40344.5340
30	2016	7	39261.9630
31	2016	8	31115.3743
32	2016	9	73410.0249
33	2016	10	59687.7450
34	2016	11	79411.9658
35	2016	12	96999.0430
36	2017	1	43971.3740
37	2017	2	20301.1334
38	2017	3	58872.3528

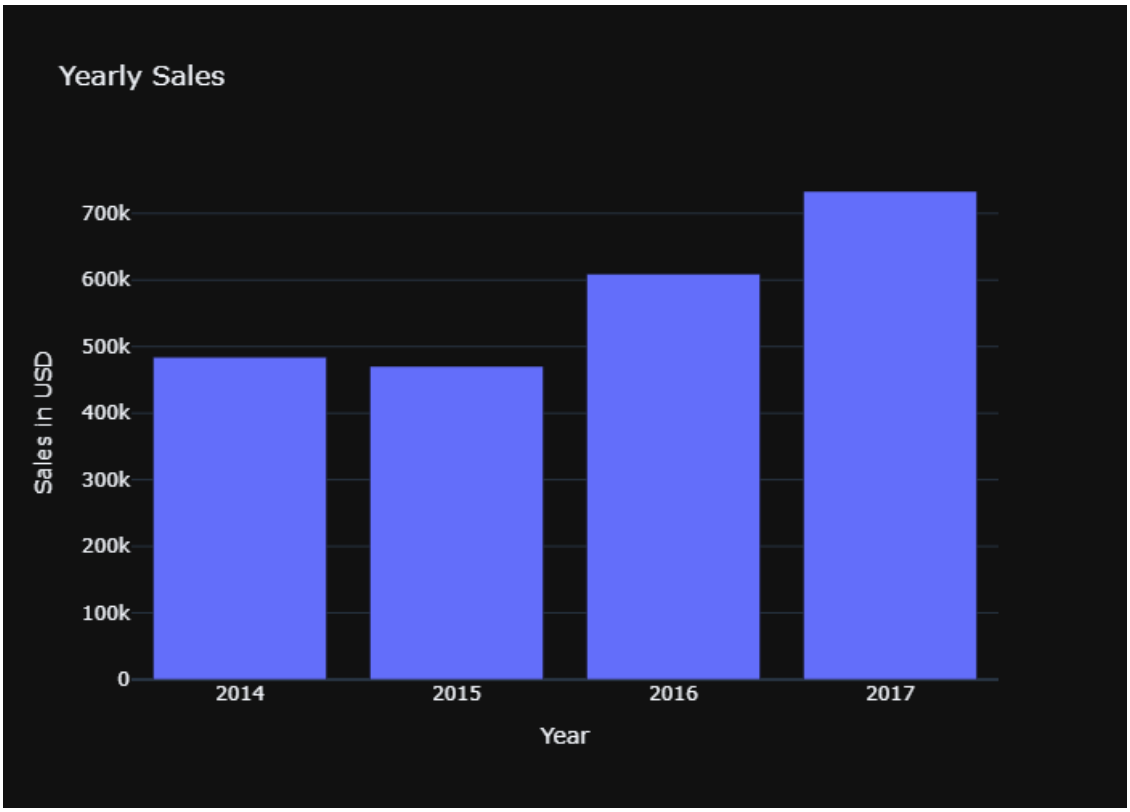
39	2017	4	36521.5361
40	2017	5	44261.1102
41	2017	6	52981.7257
42	2017	7	45264.4160
43	2017	8	63120.8880
44	2017	9	87866.6520
45	2017	10	77776.9232
46	2017	11	118447.8250
47	2017	12	83829.3188





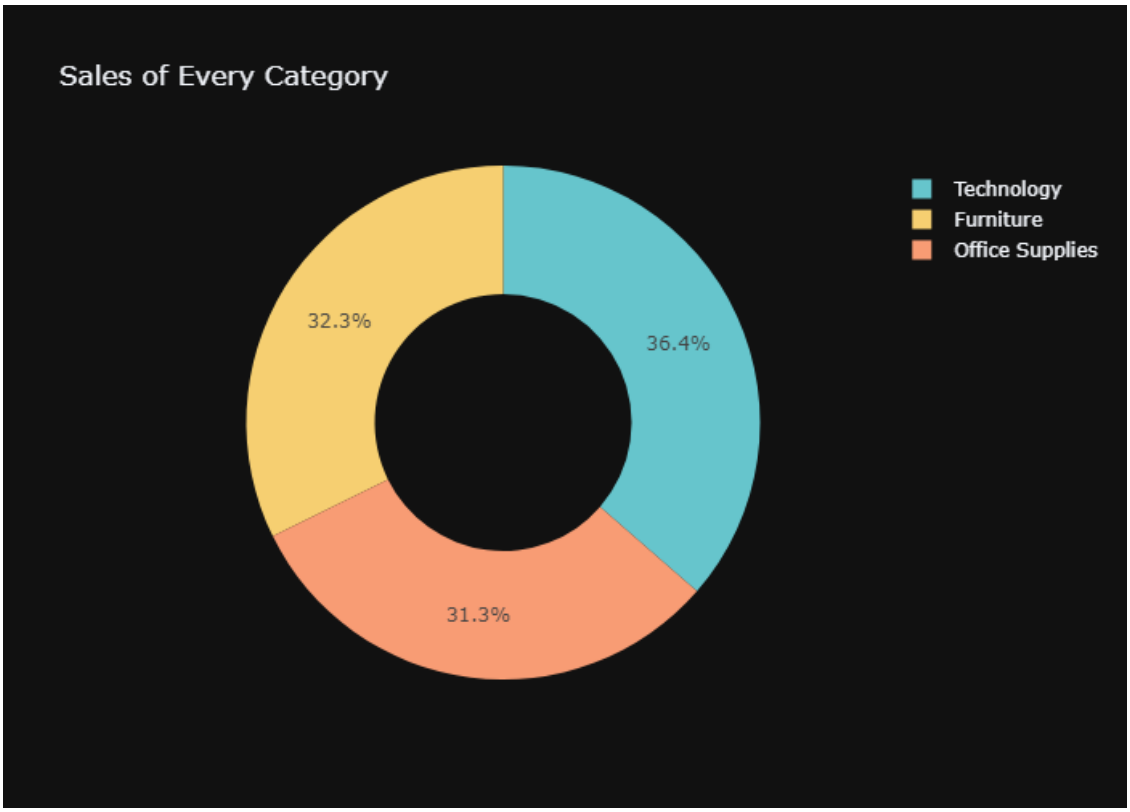
[39]:
```
# Lets check the yearly sales
yearly_sales = df.groupby('Order_Year')['Sales'].sum().reset_index()
print(yearly_sales)
# let's create a figure of yearly sales
fig = px.bar(yearly_sales,
             x='Order_Year',
             y='Sales',
             title='Yearly Sales',
             labels={'Sales':'Sales in USD', 'Order_Year':'Year'})
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='yearly_sales_bar.png')
from IPython.display import Image , display
display(Image(filename='yearly_sales_bar.png'))
```

```
   Order_Year        Sales
0        2014  484247.4981
1        2015  470532.5090
2        2016  609205.5980
3        2017  733215.2552
```



[41]:
```
# let's check the  Sales of every category
category_sales = df.groupby('Category')['Sales'].sum().reset_index()
print(category_sales)
# let's create a figure of Sales of every category
fig = px.pie(category_sales,
             values='Sales',
             names='Category',
             title='Sales of Every Category' ,
             hole=0.5 ,
             color_discrete_sequence=px.colors.qualitative.Pastel)
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='category_sales_pie.png')
from IPython.display import Image , display
display(Image(filename='category_sales_pie.png'))
```

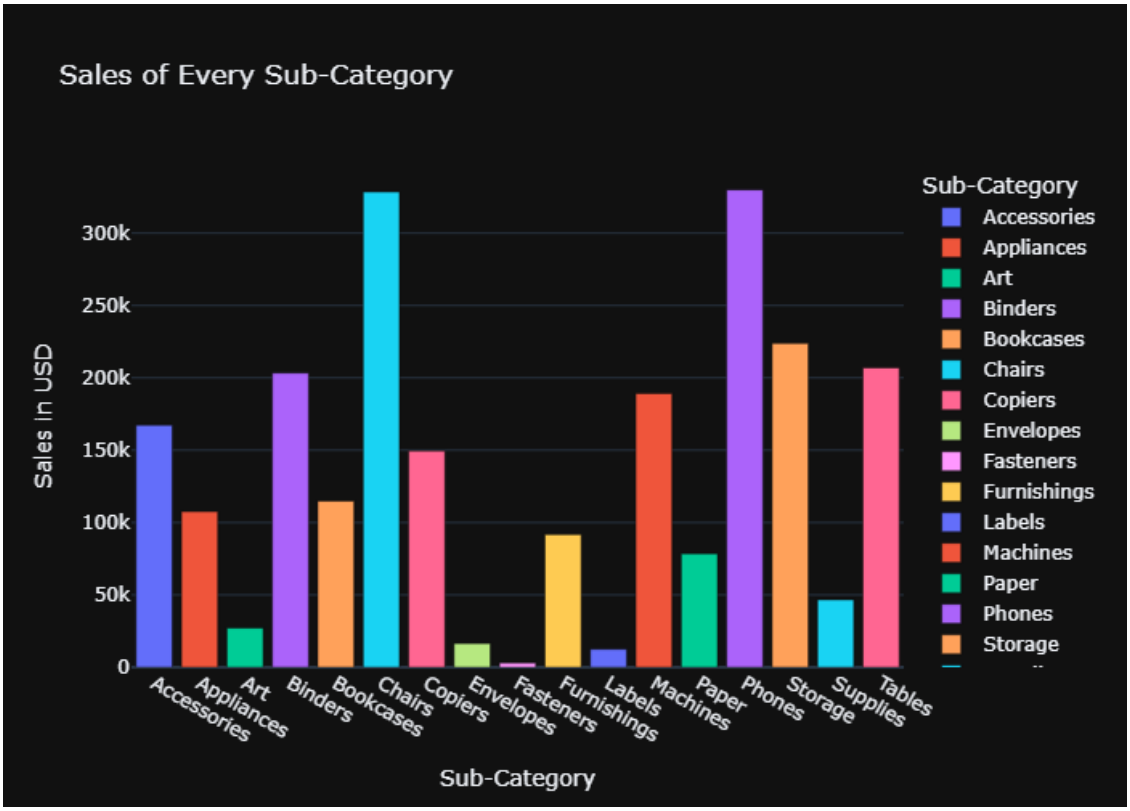
```
          Category        Sales
0        Furniture  741999.7953
1  Office Supplies  719047.0320
2       Technology  836154.0330
```



[43]:
```
# Sales of every sub-category
subcategory_sales = df.groupby('Sub-Category')['Sales'].sum().reset_index()
print(subcategory_sales)
# let's create a figure of Sales of every sub-category
fig = px.bar(subcategory_sales,
             x='Sub-Category',
             y='Sales',
             title='Sales of Every Sub-Category',
             labels={'Sales':'Sales in USD', 'Sub-Category':'Sub-Category'},
             color='Sub-Category')
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='subcategory_sales_bar.png')
from IPython.display import Image , display
display(Image(filename='subcategory_sales_bar.png'))
```

```
   Sub-Category        Sales
0   Accessories  167380.3180
1    Appliances  107532.1610
2           Art   27118.7920
```

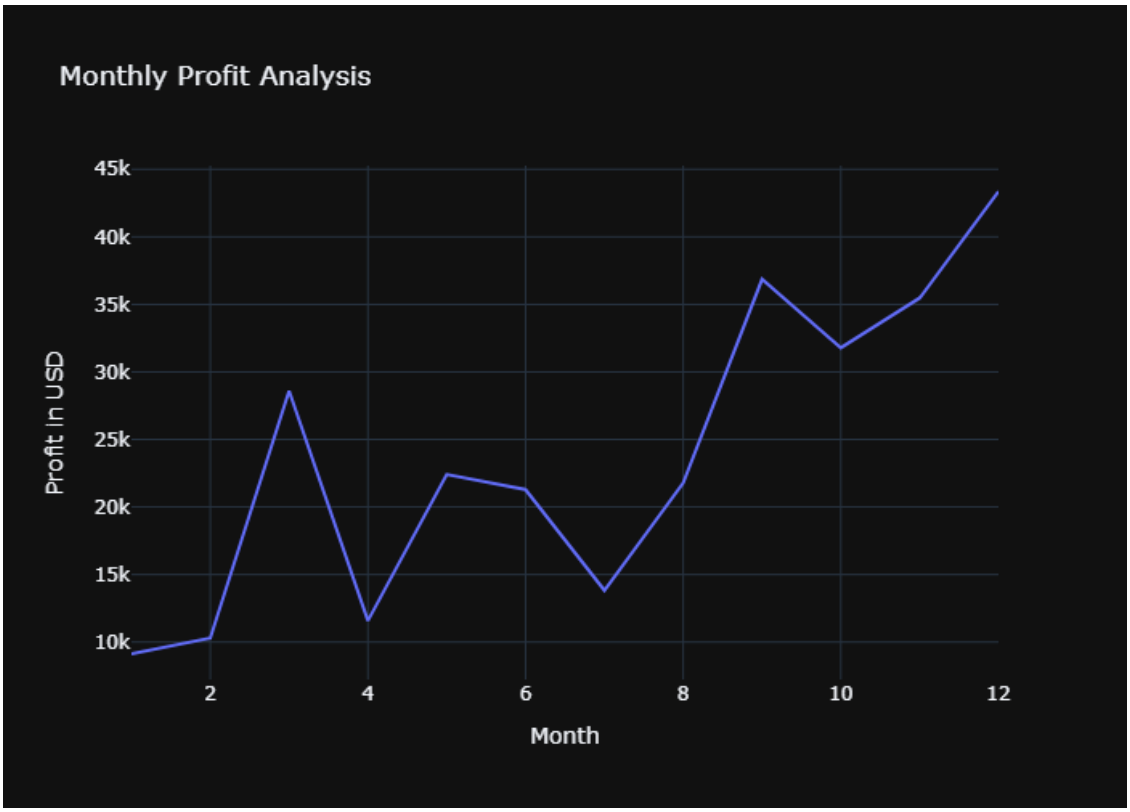
```
3       Binders  203412.7330
4     Bookcases  114879.9963
5        Chairs  328449.1030
6       Copiers  149528.0300
7     Envelopes   16476.4020
8     Fasteners    3024.2800
9   Furnishings   91705.1640
10       Labels   12486.3120
11     Machines  189238.6310
12        Paper   78479.2060
13       Phones  330007.0540
14      Storage  223843.6080
15     Supplies   46673.5380
16       Tables  206965.5320
```



[46]:
```
# Monthly profit analysis
monthly_profit = df.groupby('Order_Month')['Profit'].sum().reset_index()
print(monthly_profit)
# let's create a figure of Monthly profit analysis
fig = px.line(monthly_profit,
              x='Order_Month',
              y='Profit',
```

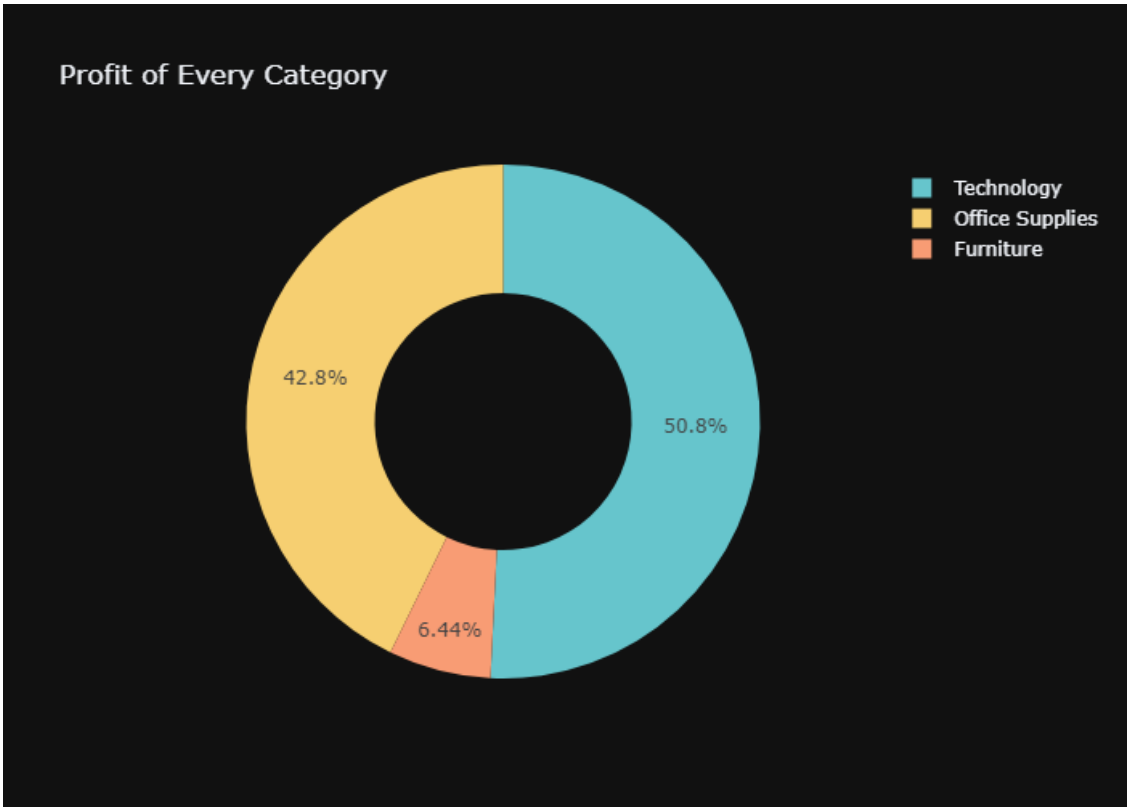
```
            title='Monthly Profit Analysis',
            labels={'Profit':'Profit in USD', 'Order_Month':'Month'} )
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='monthly_profit_line.png')
from IPython.display import Image , display
display(Image(filename='monthly_profit_line.png'))
```

	Order_Month	Profit
0	1	9134.4461
1	2	10294.6107
2	3	28594.6872
3	4	11587.4363
4	5	22411.3078
5	6	21285.7954
6	7	13832.6648
7	8	21776.9384
8	9	36857.4753
9	10	31784.0413
10	11	35468.4265
11	12	43369.1919



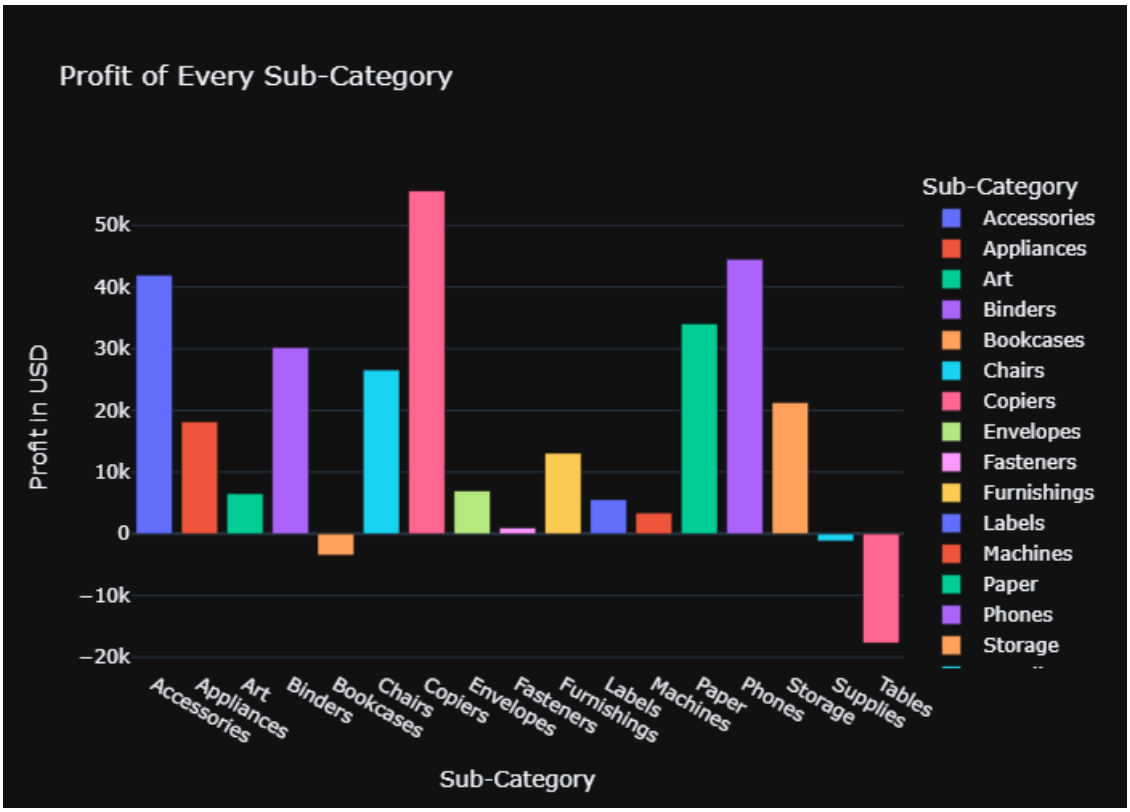
[48]:
```
# Analysis of profit of every category
category_profit = df.groupby('Category')['Profit'].sum().reset_index()
print(category_profit)
# let's create a figure of profit of every category
fig = px.pie(category_profit,
             values='Profit',
             names='Category',
             title='Profit of Every Category' ,
             hole=0.5 ,
             color_discrete_sequence=px.colors.qualitative.Pastel)
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='category_profit_pie.png')
from IPython.display import Image , display
display(Image(filename='category_profit_pie.png'))
```

```
          Category       Profit
0        Furniture   18451.2728
1  Office Supplies  122490.8008
2       Technology  145454.9481
```



[50]:
```
# Analysis of profit of every sub-category
subcategory_profit = df.groupby('Sub-Category')['Profit'].sum().reset_index()
print(subcategory_profit)
# let's create a figure of profit of every sub-category
fig = px.bar(subcategory_profit,
             x='Sub-Category',
             y='Profit',
             title='Profit of Every Sub-Category',
             labels={'Profit':'Profit in USD', 'Sub-Category':'Sub-Category'},
             color='Sub-Category')
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='subcategory_profit_bar.png')
from IPython.display import Image , display
display(Image(filename='subcategory_profit_bar.png'))
```

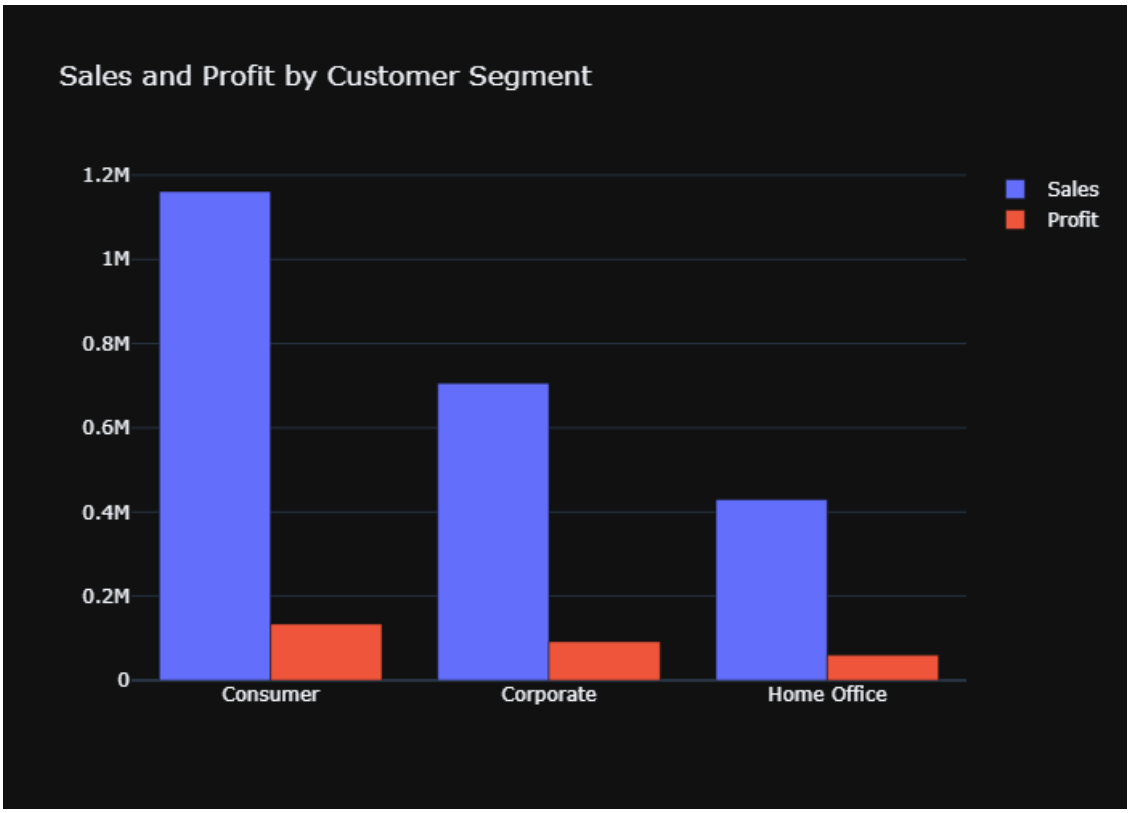
```
   Sub-Category      Profit
0   Accessories  41936.6357
1    Appliances  18138.0054
2           Art   6527.7870
3       Binders  30221.7633
4     Bookcases  -3472.5560
5        Chairs  26590.1663
6       Copiers  55617.8249
7     Envelopes   6964.1767
8     Fasteners    949.5182
9   Furnishings  13059.1436
10       Labels   5546.2540
11     Machines   3384.7569
12        Paper  34053.5693
13       Phones  44515.7306
14      Storage  21278.8264
15     Supplies  -1189.0995
16       Tables -17725.4811
```



[52]:
```
 # Analyse the sales and profit by customer segment
segment_sales_profit = df.groupby('Segment')[['Sales','Profit']].sum().
 ↪reset_index()
print(segment_sales_profit)
# let's create a figure of sales and profit by customer segment
fig = go.Figure()
fig.add_trace(go.Bar(x=segment_sales_profit['Segment'],␣
 ↪y=segment_sales_profit['Sales'], name='Sales'))
fig.add_trace(go.Bar(x=segment_sales_profit['Segment'],␣
 ↪y=segment_sales_profit['Profit'], name='Profit'))
fig.update_layout(title='Sales and Profit by Customer Segment', barmode='group')
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='segment_sales_profit_bar.png')
from IPython.display import Image , display
display(Image(filename='segment_sales_profit_bar.png'))
```

```
       Segment         Sales       Profit
0     Consumer  1.161401e+06  134119.2092
1    Corporate  7.061464e+05   91979.1340
```

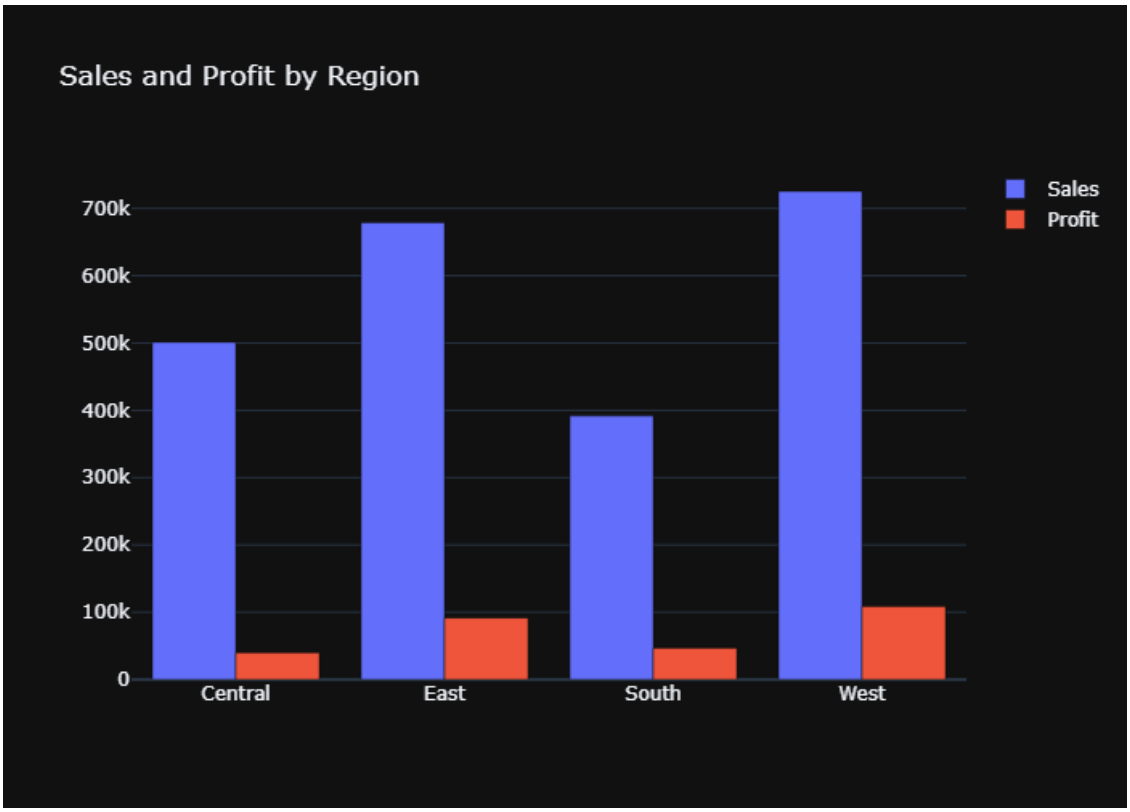
```
2  Home Office  4.296531e+05   60298.6785
```



[54]:
```
# Analyse the sales and profit by region
region_sales_profit = df.groupby('Region')[['Sales','Profit']].sum().
 ↪reset_index()
print(region_sales_profit)
# let's create a figure of sales and profit by region
fig = go.Figure()
fig.add_trace(go.Bar(x=region_sales_profit['Region'],␣
 ↪y=region_sales_profit['Sales'], name='Sales'))
fig.add_trace(go.Bar(x=region_sales_profit['Region'],␣
 ↪y=region_sales_profit['Profit'], name='Profit'))
fig.update_layout(title='Sales and Profit by Region', barmode='group')
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='region_sales_profit_bar.png')
from IPython.display import Image , display
display(Image(filename='region_sales_profit_bar.png'))
```

```
    Region         Sales       Profit
```

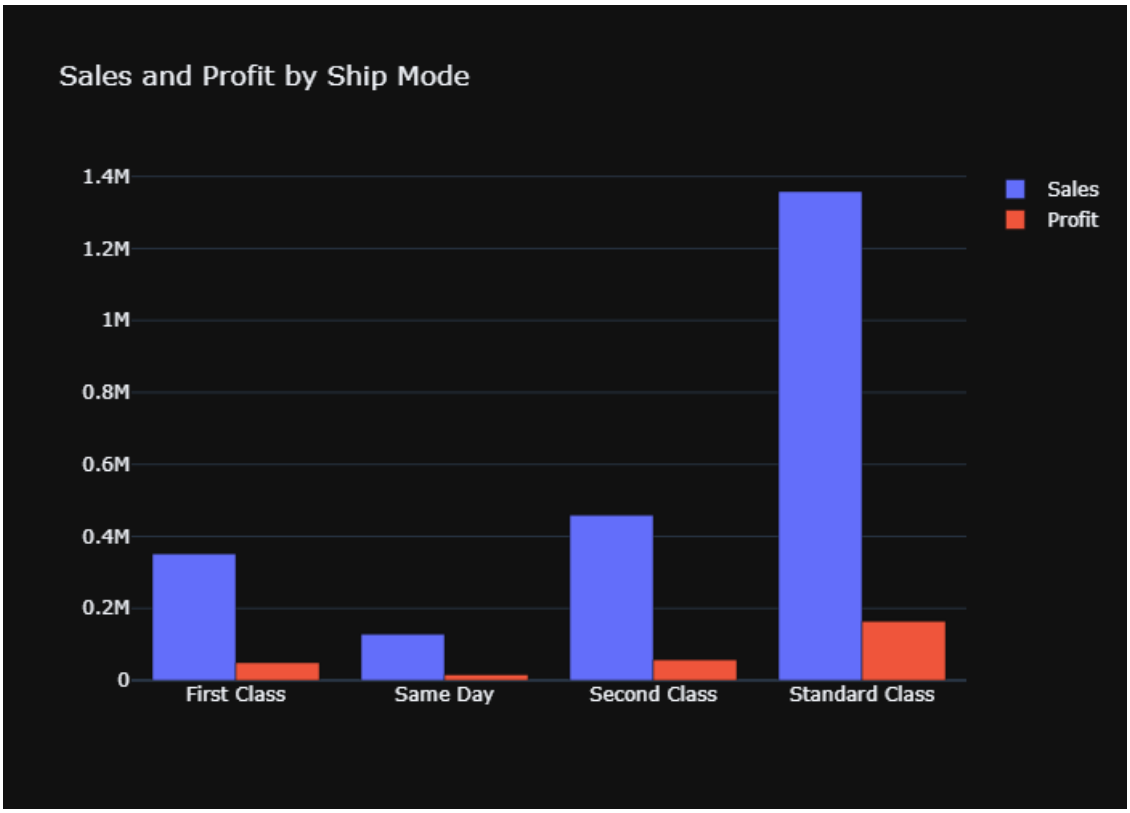
```
0  Central  501239.8908   39706.3625
1     East  678781.2400   91522.7800
2    South  391721.9050   46749.4303
3     West  725457.8245  108418.4489
```



[56]:
```
# Analyse the sales and profit by Ship Mode
shipmode_sales_profit = df.groupby('Ship Mode')[['Sales','Profit']].sum().
 ↪reset_index()
print(shipmode_sales_profit)
# let's create a figure of sales and profit by Ship Mode
fig = go.Figure()
fig.add_trace(go.Bar(x=shipmode_sales_profit['Ship Mode'],␣
 ↪y=shipmode_sales_profit['Sales'], name='Sales'))
fig.add_trace(go.Bar(x=shipmode_sales_profit['Ship Mode'],␣
 ↪y=shipmode_sales_profit['Profit'], name='Profit'))
fig.update_layout(title='Sales and Profit by Ship Mode', barmode='group')
fig.show()

# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='shipmode_sales_profit_bar.png')
```

```
from IPython.display import Image , display
display(Image(filename='shipmode_sales_profit_bar.png'))
```

```
        Ship Mode         Sales       Profit
0     First Class  3.514284e+05   48969.8399
1        Same Day  1.283631e+05   15891.7589
2    Second Class  4.591936e+05   57446.6354
3  Standard Class  1.358216e+06  164088.7875
```



[58]:
```
# Analyse the Sales ratio by region
region_sales_ratio = df.groupby('Region')['Sales'].sum().reset_index()
region_sales_ratio['Sales_Ratio'] = region_sales_ratio['Sales']/
 ↪region_sales_ratio['Sales'].sum()
print(region_sales_ratio)
# let's create a figure of Sales ratio by region
fig = px.pie(region_sales_ratio,
             values='Sales_Ratio',
             names='Region',
             title='Sales Ratio by Region' ,
             hole=0.5 ,
             color_discrete_sequence=px.colors.qualitative.Pastel)
fig.show()
```

```
# save this figure in jypiter notebook for when we export the notebook to html␣
 ↪it will be saved
# pio.write_image(fig, file='region_sales_ratio_pie.png')
from IPython.display import Image , display
display(Image(filename='region_sales_ratio_pie.png'))
```

	Region	Sales	Sales_Ratio
0	Central	501239.8908	0.218196
1	East	678781.2400	0.295482
2	South	391721.9050	0.170521
3	West	725457.8245	0.315801