In [32]:
```python
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as colors
pio.templates.default = "plotly_white"
```

In [38]:
```python
data = pd.read_csv('C:\\Users\\Admin\\Desktop\\python project\\Sample - Supers
```

In [39]:
```python
data.shape
```

Out[39]:
```
(9994, 21)
```

In [40]:
```python
data.columns
```

Out[40]:
```
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 [41]:
```python
data.head()
```

Out[41]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name
0	1	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute
1	2	CA-2016-152156	11-08-2016	11-11-2016	Second Class	CG-12520	Claire Gute
2	3	CA-2016-138688	06-12-2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff
3	4	US-2015-108966	10-11-2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell
4	5	US-2015-108966	10-11-2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell

5 rows × 21 columns

```
In [42]: data.describe()
```

Out[42]:

	Row ID	Postal Code	Sales	Quantity	Discount	
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399

```
In [43]: data.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
```

CONVERTING DATA TYPE IN DATE COLUMNS from OBJECT TO DATETIME TYPE

In [45]:
```
data['Order Date'] = pd.to_datetime(data['Order Date'], format='mixed', dayfir
```

In [46]:
```
data['Ship Date'] = pd.to_datetime(data['Ship Date'], format='mixed', dayfirst
```

In [47]:
```
data.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
```

In [48]:
```
data.head()
```

Out[48]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell

5 rows × 21 columns

ADDING NEW COLUMNS - YEAR , MONTH,DAY,WEEK

In [64]:

```
# Convert 'Order Date' to datetime if not already done
data['Order Date'] = pd.to_datetime(data['Order Date'], errors='coerce')

# Extract Year, Month, and Day of the Week
data['Order_Year'] = data['Order Date'].dt.year
data['Order_Month'] = data['Order Date'].dt.month
data['Order day of Week'] = data['Order Date'].dt.dayofweek  # Returns 0 (Monc
```

In [66]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 26 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
 21  Order_Year          9994 non-null   int32
 22  Order_Month         9994 non-null   int32
 23  Order_DayName       9994 non-null   object
 24  Day_Order           9994 non-null   int32
 25  Order day of Week   9994 non-null   int32
dtypes: datetime64[ns](2), float64(3), int32(4), int64(3), object(14)
memory usage: 1.8+ MB
```

```
In [104…  data.head()
```

Out[104…

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell

5 rows × 26 columns

MONTHLY SALES ANALYSIS

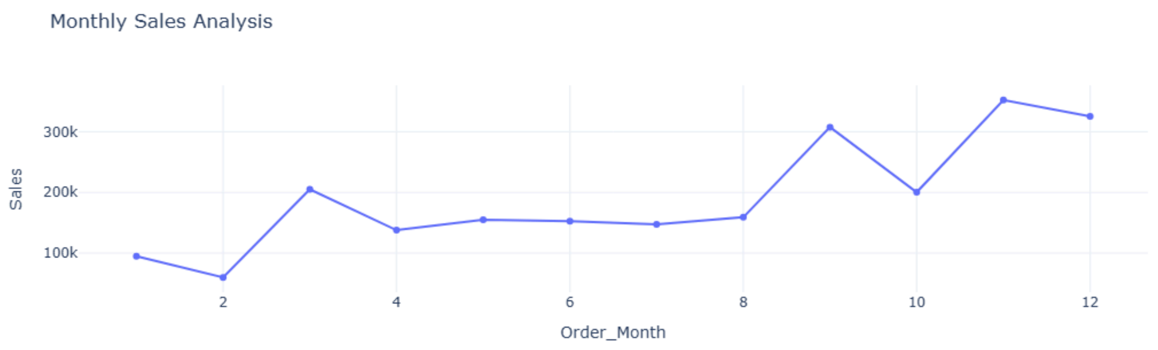
In [67]:
```
Sales_by_month = data.groupby('Order_Month')['Sales'].sum().reset_index()
```

In [68]:
```
Sales_by_month
```

Out[68]:

	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

In [69]:
```
fig = px.line(Sales_by_month,
              x='Order_Month',
              y='Sales',
              title='Monthly Sales Analysis',
              markers=True)

fig.show()
```



CATEGORY WISE SALES

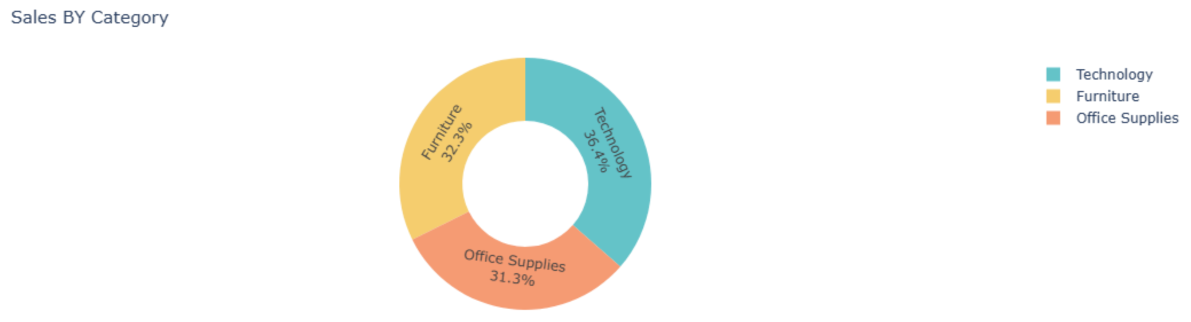
In [70]:
```
Sales_by_Category = data.groupby('Category')['Sales'].sum().reset_index()
```

In [71]:
```
Sales_by_Category
```

```
Out[71]:
```

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

```
In [72]: fig =fig = px.pie(Sales_by_Category,
                      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 BY Category',title_font=dict(size=15))

         fig.show()
```



SALES ANALYSIS BY SUB CATEGORY

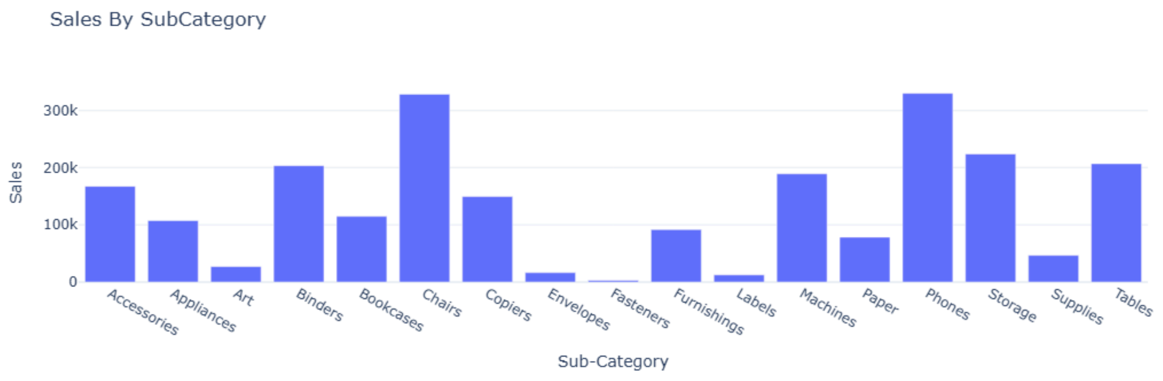
```
In [73]: Sales_by_subcategory=data.groupby('Sub-Category')['Sales'].sum().reset_index()
```

```
In [74]: Sales_by_subcategory
```

Out[74]:

	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

In [75]:
```
fig=px.bar(Sales_by_subcategory,
           x='Sub-Category',
           y='Sales',
           title='Sales By SubCategory')
fig.show()
```



SALES BY CUSTOMER SEGEMNT

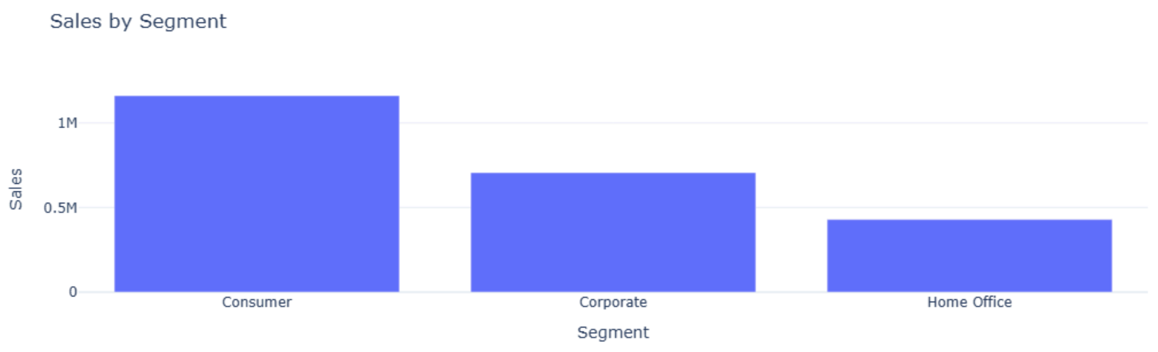
In [76]:
```
sales_by_customersegment =data.groupby('Segment')['Sales'].sum().reset_index()
```

In [77]:
```
sales_by_customersegment
```

Out[77]:

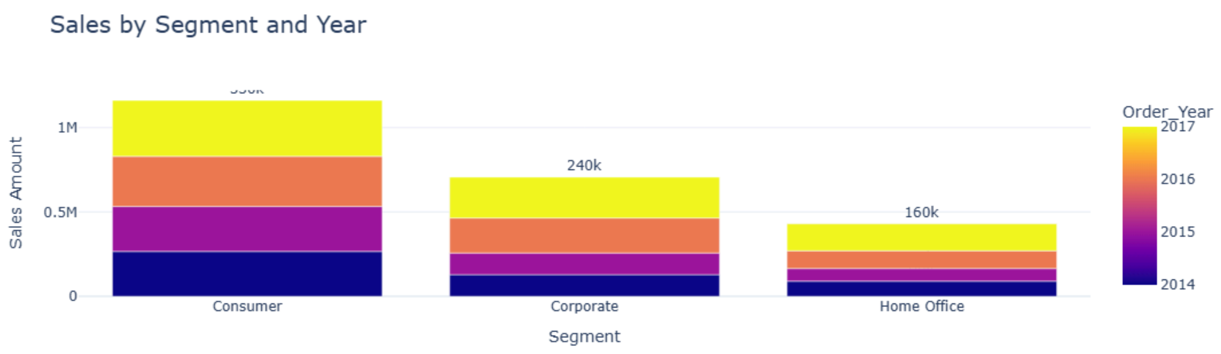
	Segment	Sales
0	Consumer	1.161401e+06
1	Corporate	7.061464e+05
2	Home Office	4.296531e+05

In [78]:
```
fig=px.bar(sales_by_customersegment,
           x='Segment',
           y='Sales',
           title='Sales by Segment')
fig.show()
```



SALES BY SEGMENT & YEAR

In [80]:
```
sales_by_segment_year = data.groupby(['Order_Year', 'Segment'])['Sales'].sum()

fig = px.bar(sales_by_segment_year,
             x='Segment',
             y='Sales',
             color='Order_Year',
             barmode='group',
             title='Sales by Segment and Year',
             text='Sales',
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
fig.update_layout(title_font=dict(size=20), yaxis_title='Sales Amount', xaxis_
fig.show()
```



MONTHLY PROFIT ANALYSIS

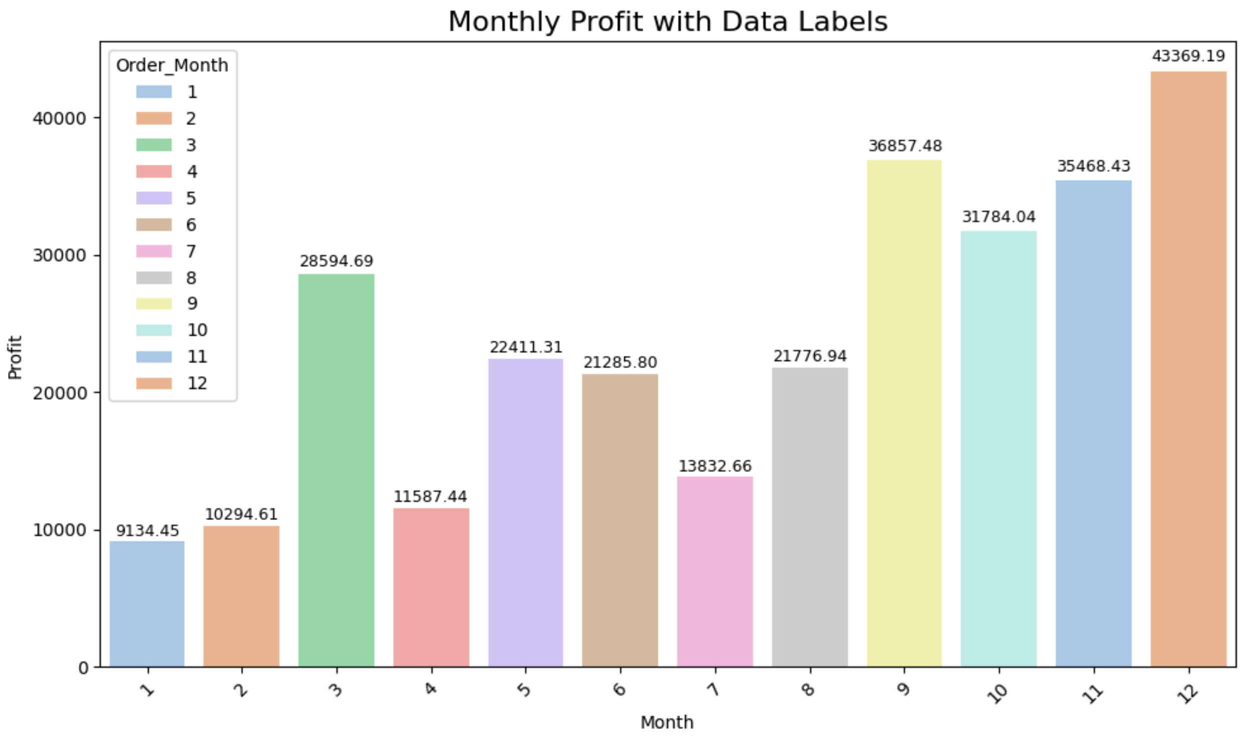
In [81]:
```
profit_month=data.groupby('Order_Month')['Profit'].sum().reset_index()
```

In [82]:
```
profit_month
```

Out[82]:

	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

In [85]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))

ax = sns.barplot(x='Order_Month', y='Profit', hue='Order_Month', data=profit_m

# Add data labels correctly
for i, row in enumerate(profit_month.itertuples()):
    ax.text(i, row.Profit + (row.Profit * 0.01), f"{row.Profit:.2f}", ha='cent

# Titles and labels
plt.title('Monthly Profit with Data Labels', fontsize=16)
plt.xlabel('Month')
plt.ylabel('Profit')
plt.xticks(rotation=45)  # Rotate labels if needed
plt.tight_layout()
plt.show()
```



PROFIT BY CATEGORY

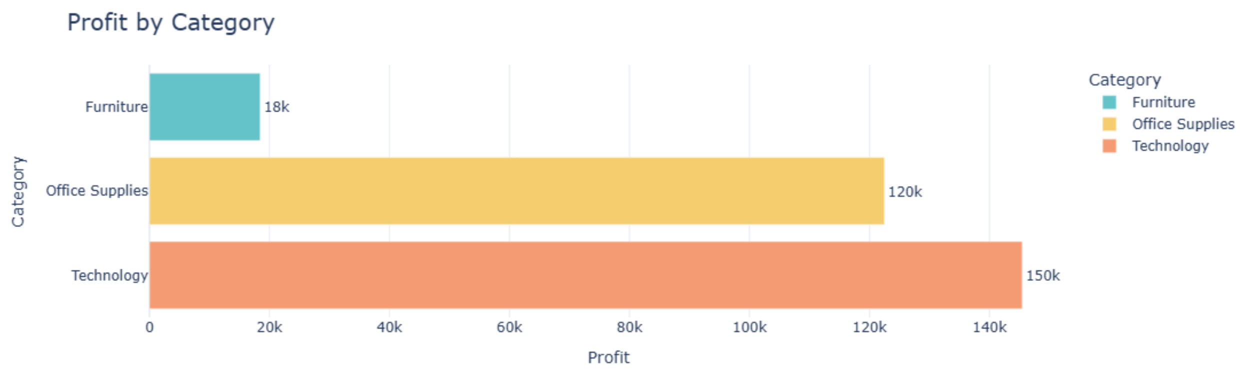
In [86]:
```
Profit_by_Category=data.groupby('Category')['Profit'].sum().reset_index()
```

In [87]:
```
Profit_by_Category
```

Out[87]:

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

In [88]:
```
fig = px.bar(Profit_by_Category,
             x='Profit',
             y='Category',
             orientation='h',
             color='Category',
             text='Profit',
             color_discrete_sequence=px.colors.qualitative.Pastel)

fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
fig.update_layout(title='Profit by Category', title_font=dict(size=20))
fig.show()
```



PROFIT BY SUBCATEGORY

In [89]:
```
Profit_by_Subcategory=data.groupby('Sub-Category')['Profit'].sum().reset_index
```

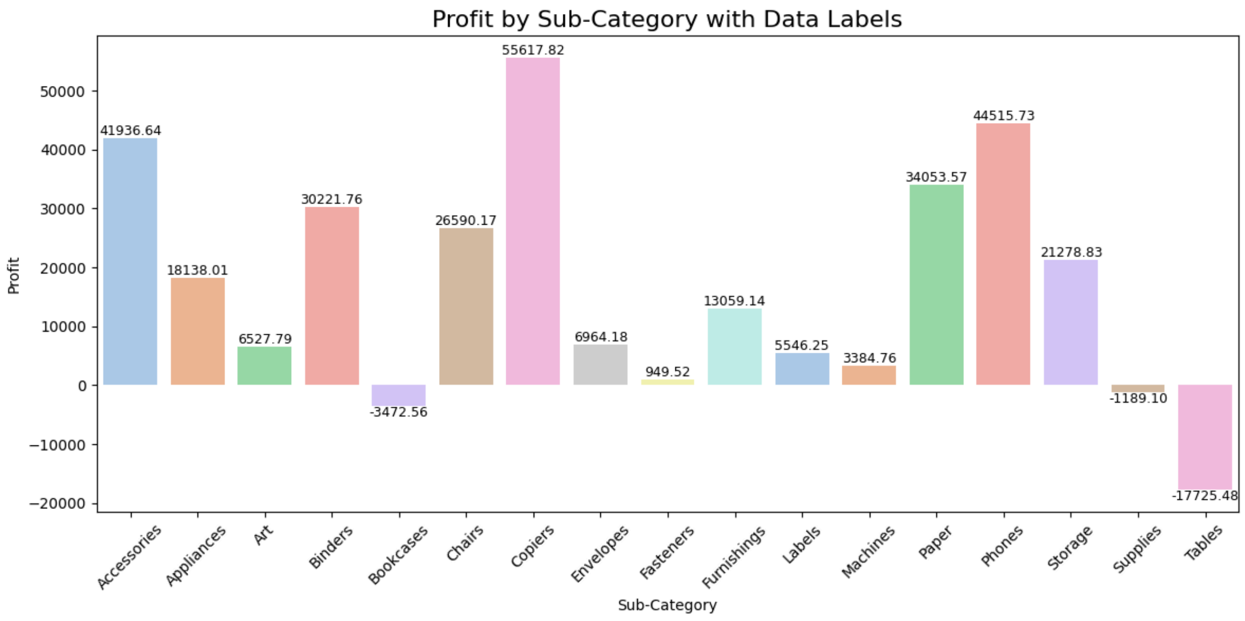
In [90]:
```
Profit_by_Subcategory
```

Out[90]:

	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

In [91]:

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
ax = sns.barplot(data=Profit_by_Subcategory, x='Sub-Category', y='Profit',
                 hue='Sub-Category', palette='pastel', legend=False)
plt.xticks(rotation=45)
for container in ax.containers:
    ax.bar_label(container, fmt='%.2f', label_type='edge', fontsize=9)

plt.title('Profit by Sub-Category with Data Labels', fontsize=16)
plt.xlabel('Sub-Category')
plt.ylabel('Profit')
plt.tight_layout()
plt.show()
```



PROFIT BY CUSTOMER SEGEMNT

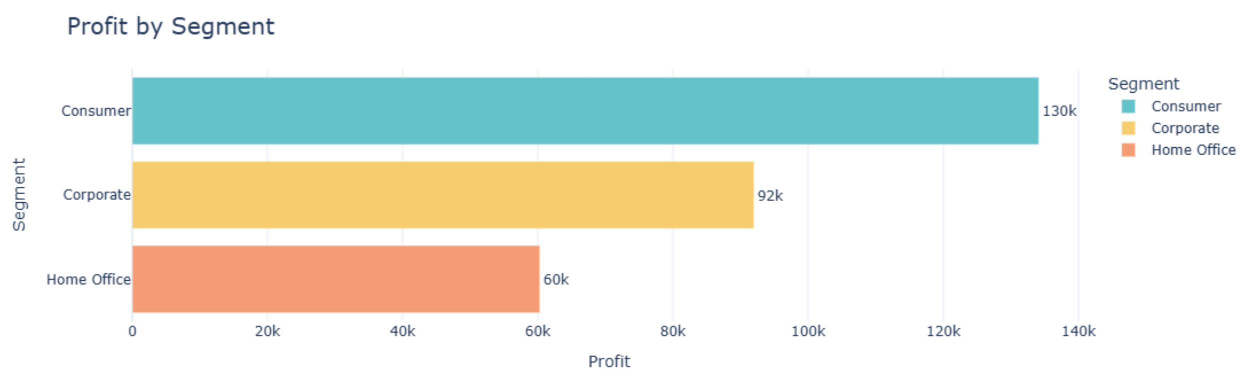
In [93]:
```
profit_by_segment= data.groupby('Segment')['Profit'].sum().reset_index()
```

In [94]:
```
profit_by_segment
```

Out[94]:

	Segment	Profit
0	Consumer	134119.2092
1	Corporate	91979.1340
2	Home Office	60298.6785

In [95]:
```
fig = px.bar(profit_by_segment,
             x='Profit',
             y='Segment',
             orientation='h',
             color='Segment',
             text='Profit',
             color_discrete_sequence=px.colors.qualitative.Pastel)

fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
fig.update_layout(title='Profit by Segment', title_font=dict(size=20))
fig.show()
```



SEGMENT & YEAR WISE PROFIT

In [96]:
```
profit_by_segment_year = data.groupby(['Order_Year', 'Segment'])['Profit'].sum
```

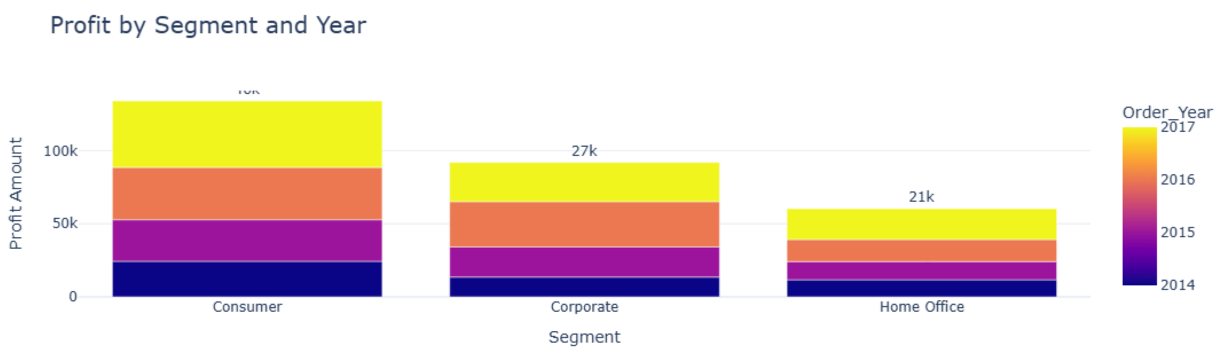
In [97]:
```
profit_by_segment_year
```

Out[97]:

	Order_Year	Segment	Profit
0	2014	Consumer	24319.8504
1	2014	Corporate	13513.2769
2	2014	Home Office	11710.8468
3	2015	Consumer	28460.1665
4	2015	Corporate	20688.3248
5	2015	Home Office	12470.1124
6	2016	Consumer	35770.9532
7	2016	Corporate	30995.1690
8	2016	Home Office	15029.0521
9	2017	Consumer	45568.2391
10	2017	Corporate	26782.3633
11	2017	Home Office	21088.6672

In [98]:
```
fig = px.bar(profit_by_segment_year,
             x='Segment',
             y='Profit',
             color='Order_Year',
             barmode='group',
             title='Profit by Segment and Year',
             text='Profit',
             color_discrete_sequence=px.colors.qualitative.Set2)
```

```
fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
fig.update_layout(title_font=dict(size=20), yaxis_title='Profit Amount', xaxis
fig.show()
```



PROFIT & SALES BY SEGMENT

In [99]:
```
sales_profit_by_segment = data.groupby('Segment').agg({'Sales': 'sum', 'Profit
```

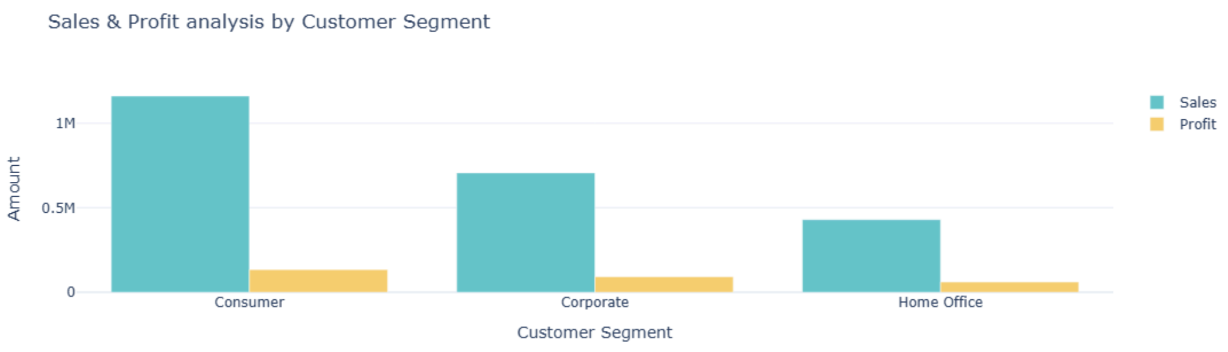
In [100…
```
sales_profit_by_segment
```

Out[100…

	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

In [101…
```
color_palette=colors.qualitative.Pastel

fig=go.Figure()

fig.add_trace (go.Bar(
    x=sales_profit_by_segment['Segment'],
    y=sales_profit_by_segment['Sales'],
    name='Sales',
    marker_color = color_palette[0]))


fig.add_trace (go.Bar(
    x=sales_profit_by_segment['Segment'],
    y=sales_profit_by_segment['Profit'],
    name='Profit',
    marker_color = color_palette[1]))

fig.update_layout(title ='Sales & Profit analysis by Customer Segment',
                  xaxis_title='Customer Segment',
                  yaxis_title='Amount')
```

```
fig.show()
```



SALES TO PROFIT RATIO

```
In [102…  sales_profit_by_segment=data.groupby('Segment').agg({'Sales':'sum','Profit':'s
          sales_profit_by_segment['Sales_to_Profit_Ratio']=sales_profit_by_segment['Sale
```

```
In [103…  print(sales_profit_by_segment[['Segment','Sales_to_Profit_Ratio']])
```

```
        Segment  Sales_to_Profit_Ratio
0      Consumer               8.659471
1     Corporate               7.677245
2   Home Office               7.125416
```

```
In [ ]:
```